НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Прикладний програмний інтерфейс для системи продажу автомобілів»

Виконав:
студент IV курсу, групи ІО-63
Пилип'юк Дмитро Олександрович _____

Керівник:
ст. викладач
Алещенко Олексій Вадимович _____

Консультант нормоконтроль:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Рецензент:
асистент
Радеченко Костянтин Олександрович
_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Пилип'юку Дмитру Олександровичу

1. Тема проєкту «Прикладний програмний інтерфейс для системи продажу автомобілів», керівник проєкту Алещенко Олексій Вадимович, ст.викладач, затверджені наказом по університету від «07» травня 2020 р. №1081-С.

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту *Технічна документація. Теоретичні та статистичні дані. Прикладний програмний інтерфейс. Демонстраційний клієнтський графічний інтерфейс. База даний MongoDB. Середовище розробки IntelliJ IDEA, Java. Додаток для відправки запитів Postman.*

4. Зміст пояснювальної записки *Аналіз предметної області, дослідження методики побудови прикладних програмних інтерфейсів за принципом REST, розробка прикладного програмного інтерфейсу та демонстраційного інтерфейсу.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) функціональна схема, схема взаємодії, структурна схема.

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*30.03.2020*	
3	*Розробка архітектури та загальної структури систем*	*10.04.2020*	
4	*Розробка структур окремих підсистем*	*22.04.2020*	
5	*Програмна реалізація системи*	*04.05.2020*	
6	*Оформлення пояснювальної записки*	*17.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*	*16.06.2020*	

Студент Дмитро ПИЛИП'ЮК

Керівник Олексій АЛЕЩЕНКО

Анотація

В бакалаврській дипломній роботі реалізовано прикладний програмний інтерфейс для системи продажу автомобілів.

Програма дозволяє користувачеві виконувати базові операції в системі продажу автомобілів: створювати та редагувати оголошення з продажу транспортного засобу, знаходити оголошення по заданим параметрам, дізнатись вартість розмитнення авто тощо. Програмний продукт був створений на мові програмування Java використанням фреймворку Spring Boot у візуальному середовищі програмування IntelliJ IDEA.

Для візуалізації, вводу та виводу даних використовується власний прототип клієнтського додатку або додаток для відправки HTTP-запитів Postman. Дані, які потрібні для роботи системи зберігаються в документо-орієнтованій базі даних MongoDB.

Annotation

The bachelor's thesis implements an application software interface for the car sales system.

The program allows the user to perform basic operations in the car sales system: create and edit ads for the sale of vehicles, find ads for the specified parameters, find out the cost of customs clearance of cars and more. The software product was created in the Java programming language using the Spring Boot framework in the IntelliJ IDEA visual programming environment.

For visualization, input and output of data the own prototype of the client application or the application for sending HTTP-requests of Postman is used. The data required for the system to work is stored in a document-oriented MongoDB database.

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Прикладний програмний інтерфейс для системи продажу автомобілів”

Київ – 2020 року

Технічне завдання до *дипломної* роботи

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Прикладний програмний інтерфейс для системи продажу автомобілів».

Область застосування мого пристрою: підприємства, що відносяться до категорії малого і середнього бізнесу в сфері продажу нових та вживаних авто.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки цього прикладного програмного інтерфейсу є завдання на виконання дипломного проекту «Прикладний програмний інтерфейс для системи продажу автомобілів», затвердженого кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою мого проекту є розробка прикладного програмного інтерфейсу, який може надати можливості створювати нові оголошення щодо продажу автомобілів, видаляти їх там змінювати, виконувати пошук оголошень. Всі дані, що були створені або оновлені повинні зберігатись в базі даних.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки мого проєкту служать: науково-технічна література, довідники по програмній інженерії, публікації в спеціалізованих виданнях, та публікації на тему розробки прикладних програмних інтерфейсів в мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

- Створення оголошення в базі даних після отримання HTTP-запиту від клієнта;
- Можливість маніпулювати даними, що вже збережені в базі даних: змінювати параметри, повністю видаляти оголошення;
- Можливість розширити функціонал прикладного програмного інтерфейсу завдяки правильній програмній архітектурі;

5.2. Вимоги до програмного забезпечення

- Операційна система: Linux або будь-яка UNIX-подібна ОС, Windows 8 та новіше, macOS, Solaris;
- Додаток для відправки HTTP-запитів Postman;
- Графічний інтерфейс Compass для MongoDB;
- Веб-браузер з переліку: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Opera;
- Віртуальна машина JVM.

5.3. Вимоги до апаратної частини

- Об’єм оперативної пам’яті: від 6 гігабайт.
- Процесор Intel або AMD з 2 і більше ядрами з тактовою частотою понад 2 гігагерц;
- Вільний простір на жорсткому диску або носій даних розміром 20 гігабайт.

6. ЕТАПИ РОЗРОБКИ

Етап	Дата
Вивчення літератури	30.01.2020
Складання і узгодження технічного завдання	11.02.2020
Створення модулів системи, що розробляється	14.03.2020
Тестування окремих модулів системи	10.04.2020
Доопрацювання, налагодження і виправлення помилок	04.05.2020
Оформлення документації дипломної роботи	17.05.2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “Прикладний програмний інтерфейс для системи продажу автомобілів”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ДП.467100.001 ТЗ	Технічне завдання	4	
3	А4	ДП.467100.002 ВП	Відомість проєкту	1	
4	А4	ДП.467100.003 ПЗ	Пояснювальна записка	63	
5	А3	ДП.467100.004 Д1	Схема структурна Діаграма класів	1	
6	А3	ДП.467100.005 Д2	Схема функціональна Блок-схема алгоритму	1	
7	А3	ДП.467100.006 Д3	Схема взаємодії	1	

Змн.	Арк.	№ докум.	Підпис	Дата	*ІАЛЦ.467100.002 ВП*		
Розроб.		Пилип'юк Д.О.			*Відомість дипломного проекту*	Літ.	Арк. 2 / Акрушів 1
Перевір.		Алещенко О.В.					
Н. Контр.		Сімоненко В.П.				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*	
Затверд.							

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: “Прикладний програмний інтерфейс для системи продажу автомобілів”

Київ – 2020 року

ЗМІСТ

ВСТУП

Сучасна людина користується транспортом (особисте авто, таксі, громадський транспорт) майже щодня. Багато сфер бізнесу потребують застосування транспорту, наприклад: магазини, пошта, пасажирські перевезення. Тенденції та бажання споживачів на ринку авто стрімко змінюються, а за ними й самі авто.

Ринок інформаційних технологій тісно пов’язаний з ринком транспорту, оскільки людям потрібні сучасні онлайн-платформи та мобільні додатки, де вони б змогли за лічені хвилини знайти автомобіль, який їх цікавить або швидко розмістити оголошення про продаж.

Велика кількість компаній, зокрема малий так середній бізнес, зацікавлена в розгортанні власної платформи для продажу та купівлі транспортних засобів, однак процес створення таких веб-сайтів потребує великих фінансових та кадрових ресурсів. Перед компаніями виникає значна низка питань від яких залежить успіх їх платформи та масштаб фінансових витрат на її розробку. Одним з найважливіших, з точки зору бізнесу, питанням є витрати на розміщення серверного обладнання, на якому буде розгорнута дана платформа та її обслуговування. Однак на першому місці стоїть питання розробки власного ресурсу, дизайну бізнес-логіки прикладного програмного інтерфейсу, який буде опрацьовувати клієнтські дані на стороні серверу, а також розмір інвестиції, які потрібно вкласти в створення власного програмного продукту.

Для компаній, що відносяться до малого та середнього бізнесу це може бути непідйомна з точки зору фінансів проблема. Процес дизайну та створення програмного забезпечення зазвичай є доволі повільним, оскільки сторони намагаються знайти найбільш оптимальне рішення, що буде задовольняти бажання та інтереси і клієнта, і провайдера послуг. Чим більше бізнес витратить часу та грошей на створення ресурсу, тим менше вигода його створення для компанії. Сучасний ринок не надає готових продуктових рішень

компаніям зі сфери купівлі-продажу авто та їх комплектуючих. Бізнес повинен сам вирішувати проблему створення свого додатку, замість того, щоб отримати вже готове програмне забезпечення, яке може бути розгорнутим на сервері за лічені дні і почати приносити прибуток протягом кількох тижнів.
WEB-додатки, які допомагають компаніям вести бізнес та заробляти за допомогою цих додатків гроші, називають корпоративними додатками або додатками підприємств (з англ. Enterprise Application). Сучасна архітектура корпоративних додатків має дві частини: клієнтську та серверну.

Клієнтська сторона додатку підприємств це WEB-сторінка або набір сторінок, які клієнт може спостерігати в своєму WEB-браузері та виконувати певні дії, в залежності від реалізованої логіки, використовуючи графічні елементи користувацького інтерфейсу, такі як: кнопки, поля для вводу тексту чи дати, зображення, дерева карток, скроллбари тощо. Клієнтська частина допомагає звичайному користувачеві взаємодіяти з серверною стороною додатка у простому та зрозумілому для нього вигляді. Сторона клієнта створюється за допомогою технологій, які можуть відтворюватись в сучасних браузерах. Це може бути як і звичайні HTML-сторінки, з використанням стилів CSS для оформлення і мови програмування JavaScript для динамічної зміни контенту на HTML-сторінках, так і фреймворки (англ. *Framework*, каркас, платформа, структура, інфраструктура) для створення сучасних та гнучких WEB-додатків.

Фреймворк – це набір програмних рішень, що спрощує розробку складних програмних систем, який надає розробникам певні точки розширення для реалізації власної логіки програмного продукту. Одним з найпопулярніших WEB-фреймворків сьогодні є Angular, створений під керівництвом компанії Google. Angular дозволяє розробникам швидко будувати клієнтські SPA-додатки (англ. Single Page Application).

Щодо серверної сторони додатку підприємства: вона приймає дані, що надіслав користувач на клієнтській стороні та виконує з ними певні дії, згідно

бізнес-логіки додатку. Зазвичай, серверна сторона також має базу даних, в якій зберігає певні дані про користувачів, події та процеси системи, товари та послуги, які надає підприємство. При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, API визначається набором точок прийому на стороні сервера на які приходять HTTP-повідомлення з визначеною структурою запиту та відповіді, зазвичай у форматі JSON, або менш часто в форматі XML. Сьогодні тенденція створення прикладних програмних інтерфейсів для WEB-додатків відходить від застосування Simple Object Access Protocol (SOAP) на основі WEB-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Метою даної бакалаврської роботи є створення прикладного програмного інтерфейсу (API) для системи продажу автомобілів, де буде реалізована базова бізнес-логіка для таких систем. Майбутні користувачі зможуть отримати перелік точок прийому даних від клієнта і отримувати певну відповідь від сервера в залежності від виконаної операції.

ОПИС ЗАВДАННЯ

Завдання даної бакалаврської роботи полягає в створенні прикладного програмного інтерфейсу для системи продажу автомобілів, який дозволить користувачам, які займаються бізнесом в сфері купівлі-продажу автомобілів застосувати його при створенні та введенні в експлуатацію власного веб-ресурсу, що дозволяє користувачам розміщувати оголошення про продаж автомобілів та шукати пропозиції, які їх цікавлять.

Користувач веб-ресурсу, який буде створений з використанням наданого прикладного програмного інтерфейсу повинен мати можливість створювати нові оголошення, видаляти їх, змінювати та додавати параметри, що були вказані при створенні, шукати оголошенням по заданим параметрам (ціна, модель, рік випуску, пробіг тощо). Всі дані про оголошення та користувача повинні зберігатись в базі даних у відповідному для них вигляді. При створенні, оголошення повинне пройти модерацію особою, яка має відповідні обов’язки.

Для презентації прикладного програмного інтерфейсу буде створено демонстраційний графічний інтерфейс користувача, з допомогою якого можна буде продемонструвати частину функціоналу прикладного програмного інтерфейсу.

Додатковими функціями прикладного програмного інтерфейсу буде можливість порахувати можливість розмитнення автомобіля, який був доставлений в Україну з іншої держави. Формули та правила обчислень відповідатимуть чинному законодавству України про розмитнення транспортних засобів.

РОЗДІЛ 1
ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА НАПРЯМКІВ ДОСЛІДЖЕННЯ

1.1 Визначення прикладного програмного інтерфейсу

Прикладний програмний інтерфейс (англ. Application Programming Interface, скор. API) – це сукупність засобів та правил, що дають можливість взаємодіяти між окремими складовими програмного забезпечення, або між програмним та апаратним забезпеченням. API надає розробнику засоби та компоненти для швидкого створення програмного забезпечення.

Прикладний програмний інтерфейс може бути створений для:

- Веб-систем;
- Операційних систем;
- Баз даних;
- Апаратного забезпечення;
- Програмних бібліотек;
- Взаємодії з вже створеним програмним продуктом.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу. Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх. У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.[1]

Часто в розробників початківців виникає проблема визначення різниці між програмною бібліотекою та прикладним програмним інтерфейсом. API, як правило пов'язаний з програмними бібліотеками, тобто він може використовувати їх у своїй реалізації. Однак, прикладний програмний інтерфейс описує і визначає певну поведінку і дає можливість її перевизначити, а програмна бібліотека – це лише реалізація певного набору правил та поведінки. Прикладний програмний інтерфейс може бути основою для програмних платформ. Щоб отримати доступ до поведінки, що вмонтована в платформу по замовчанню, потрібно розширити зміст класів платформи або додати свою реалізацію певного функціоналу. Зазвичай, сучасні програмні платформи беруть потік виконання програми під свій контроль.

1.2 Аналіз існуючих рішень

Сьогодні сфера купівлі-продажу автомобілів є дуже популярною в Україні, оскільки є можливість доставити авто з закордону і придбати його за доступну ціну. Оскільки, ринок авто, зокрема вживаних, виріс, то й виріс попит на пошук автомобілів та розміщення оголошень про їх продаж.

1.2.1 AUTO.RIA

Найпопулярнішою платформою з продажу вживаних та нових авто є «AUTO.RIA». Цей ресурс є лідером автомобільної інтернет-торгівлі вже кілька років поспіль. Зокрема, «AUTO.RIA» єдиний ресурс, який надає свій прикладний програмний інтерфейс у відкритому доступі з можливістю відсилати тисячу запитів на годину безкоштовно. Однак, API надає доступ до загальної бази платформи, тобто ви можете виконувати операції лише над даними, що відображаються на сайті.

Для того, щоб користуватись прикладним програмним інтерфейсом AUTO.RIA, необхідно перейти на веб-сторінку https://developers.ria.com/ та зареєструватись. Також тут можна знайти детальну документацію про те, як користуватись даним прикладним програмним інтерфейсом.

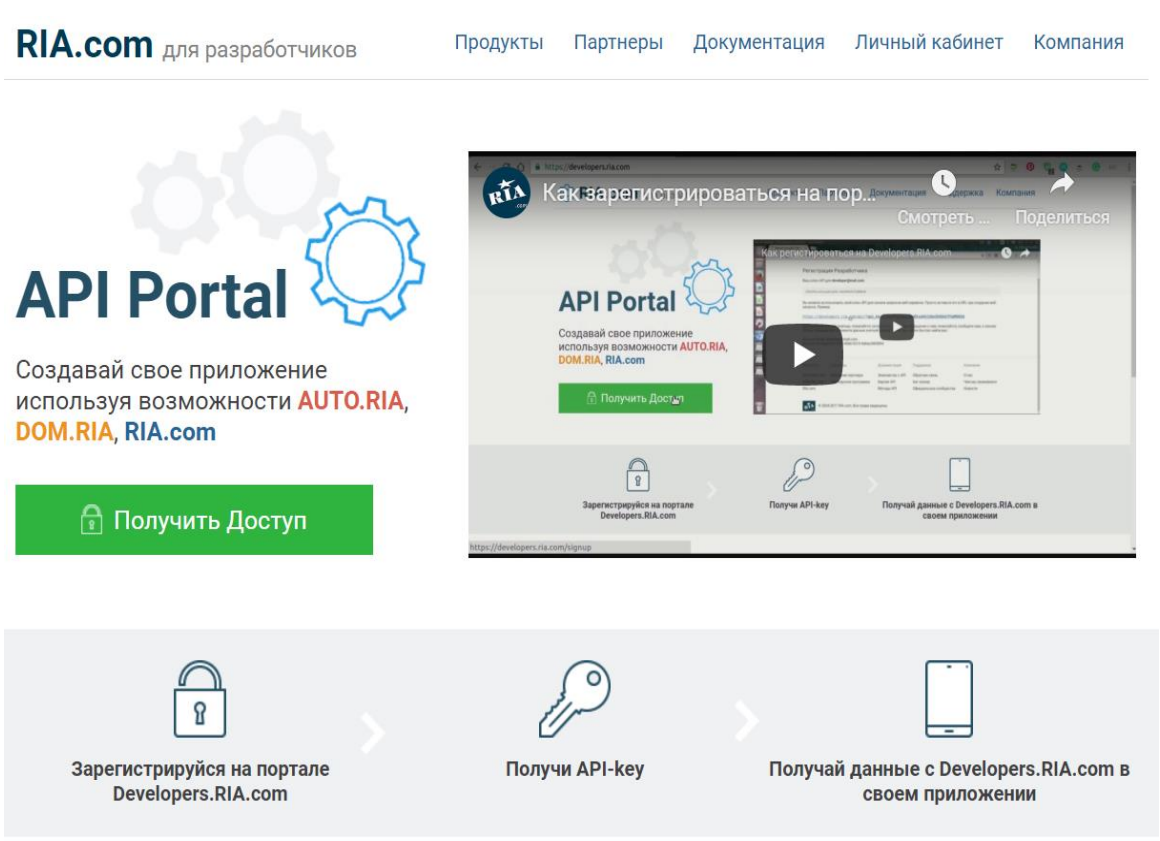


Рисунок 1.1. Головне вікно порталу для розробників AUTO.RIA

Після реєстрації, в особистому кабінеті користувача буде доступний API-ключ, який необхідно використовувати при роботі з прикладним програмним інтерфейсом AUTO.RIA, щоб пройти аутентифікацію.

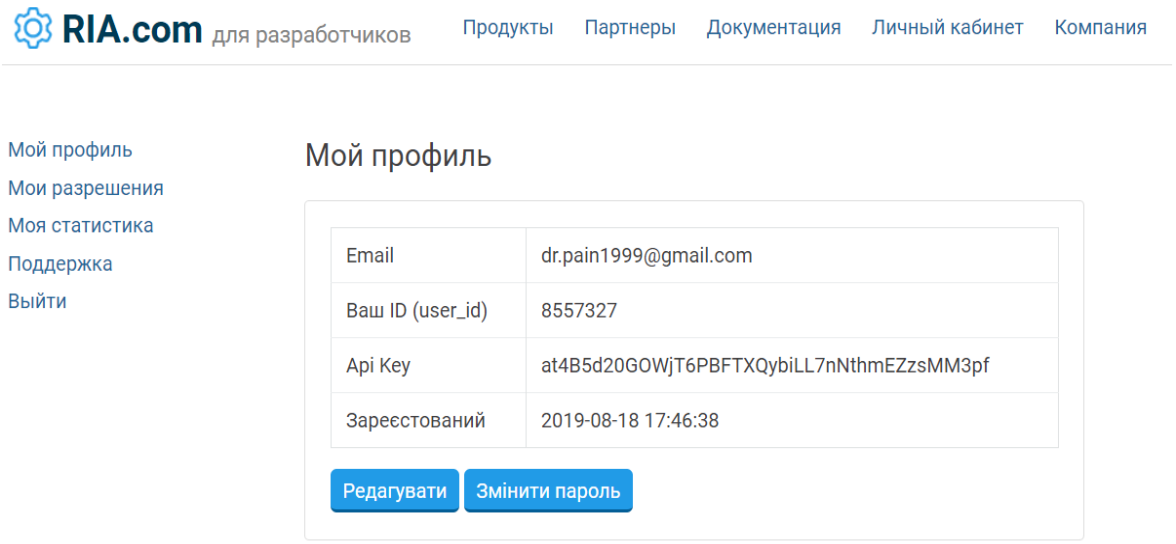


Рисунок 1.2 Особистий кабінет розробника

Тепер API-ключ можна використати, щоб отримати дані, використовуючи прикладний програмний інтерфейс AUTO.RIA. Погодинні лічильники API-ключа анулюються на постійній основі. Наприклад, якщо користувач виконує 500 запитів о 10:15 і 500 запитів о 10:25, то ключ буде тимчасово заблокований і будь-які дії з API будуть недоступні. Блок буде знятий через годину, о 11:15, після чого користувач зможе відправити ще 500 запитів. Якщо ключ перевищує дозволений ліміт запитів, то користувач отримає у відповідь HTTP-статус 429 (занадто багато запитів). Якщо користувач хоче збільшити кількість дозволених запитів до API для своїй певних потреб, йому рекомендують зв'язатись з представниками AUTO.RIA та обговорити комерційну співпрацю.

Прикладний програмний інтерфейс AUTO.RIA можна використовувати як розробник, використовуючи API-ключ і працювати з ним безпосередньо, або ж використовувати веб-сайт AUTO.RIA як звичайний користувач.

Звернутись до прикладного програмного інтерфейсу веб-додатку можна звернутись кількома способами, зокрема:

- Поле для URL веб-браузера;
- Утиліта командного рядка «cUrl»;
- Додаток для відправлення HTTP-запитів «Postman».

Використаємо Postman, щоб відправити запит до API AUTO.RIA, оскільки цей додаток має зрозумілий користувачу інтерфейс та широкий функціонал для налаштування та відправки HTTP-запитів. Також Postman може конвертувати відповідь сервера у різні формати, що допомагає відобразити дані у зручному для читання вигляді.

Виконаємо запит, який дозволяє шукати оголошення по заданим параметрам, тобто пошук. Для того, щоб звернутися до пошуку в API AUTO.RIA, необхідно виконати HTTP-запит з методом GET такого вигляду: https://developers.ria.com/auto/search?api_key=YOUR_API_KEY&PARAMETERS

Або:

curl -i -g -

X GET https://developers.ria.com/auto/search?api_key=YOUR_API_KEY&PARAMETERS

при використанні утиліти cURL.

Запит складається із:

- search – назва методу прикладного програмного інтерфейсу, до якого бажають звернутись.
- API_KEY – ключ, який був отриманий в особистому кабінеті.
- PARAMETERS – вхідні параметри, за якими буде здійснюватися пошук потрібних оголошень. Параметри для GET-запиту потрібно вказувати як послідовність пар «ключ-значення», що розділені амперсандом. Детальний список дозволених параметрів для пошуку вказаний на сайті з документацією до API AUTO.RIA.

Відповідь на такий запит буде отримана в форматі JSON.

Припустимо, пошук відбувається по заданим параметрам:

- Легкові машини (category_id = 1)
- Кузов Седан, Універсал (bodystyle [0] = 3 & bodystyle [4] = 2)
- Японія (brandOrigin [0] = 276)
  - Toyota (marka_id [0] = 79)
  - Всі моделі (model_id [0] = 0)
  - Рік випуску від 2010 до 2017р. (S_yers [0] = 2010 & po_yers [0] = 2017)
- Німеччина (brandOrigin [1] = 392)
  - Volkswagen (marka_id [1] = 84)
  - Всі моделі (model_id [1] = 0)
  - Рік випуску від 2012 з 2016р. (S_yers [1] = 2012 & po_yers [1] = 2016)
- Ціна від 1000 до 60000 (price_ot = 1000 & price_do = 60000)
  - Ціна вказана в доларах США (currency = 1)

- Можливий торг (auctionPossible = 1)
- Можливий обмін на нерухомість (with_real_exchange = 1)
- Можливий обмін на автомобіль (with_exchange = 1)
  - Марка автомобіля будь-яка (exchange_filter [marka_id] = 0)
  - Модель автомобіля будь-яка (exchange_filter [model_id] = 0)
- Область
  - Вінницька - пошук по всіх містах цієї області (state [0] = 1 & city [0] = 0)
  - Житомирська - пошук по всіх містах цієї області (state [1] = 2 & city [1] = 0)
  - Київська - пошук по всіх містах цієї області (state [2] = 10 & city [2] = 0)
- Чи не відображати авто які находяться не в Україні (abroad = 2)
- Чи не відображати нерозмитнені авто (custom = 1)
- Гаражне зберігання (auto_options [477] = 477)
- Паливо
  - Бензин (type [0] = 1)
  - Дизель (type [1] = 2)
  - Газ / бензин (type [3] = 4)
  - Електро (type [5]: 6)
- КПП
  - Ручна / Механіка (gearbox [0] = 1)
  - Автомат (gearbox [1] = 2)
  - Типтронік (gearbox [2] = 3)
- Обсяг 1.4 - 3.2 л. (EngineVolumeFrom = 1.4 & engineVolumeTo = 3.2)
- Потужність 90 - 250 (powerFrom = 90 & powerTo = 250)
  - Одиниця виміру потужності - к.с (power_name = 1)

Тільки з фото (with_photo = 1)

У випадку, якщо запит із заданими параметрами був успішно виконаний, у відповідь буде отримано HTTP-статус 200 «OK».

Використаємо Postman, щоб відправити описаний вище HTTP-запит та отримати відповідь.

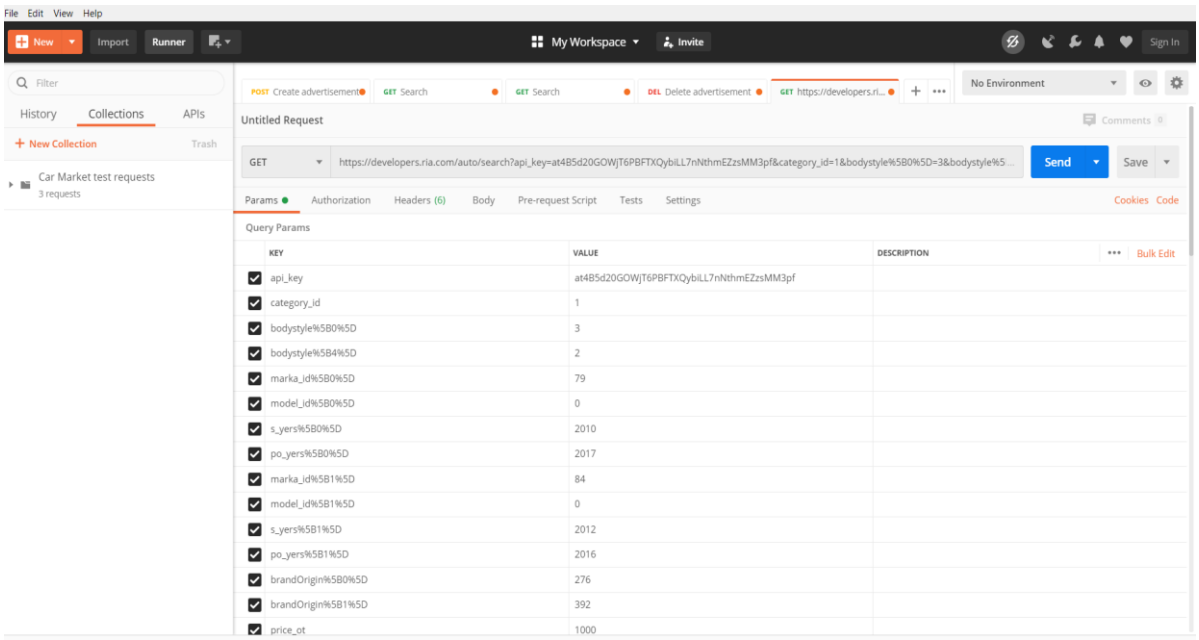


Рисунок 1.3. Конфігурація запиту в Postman

Зазначимо, що максимальна кількість відображених id оголошень за один запит дорівнює 100. У випадку, якщо результат перевищує це число, можна додати значення параметру page (порядковий номер сторінки), з допомогою якого можна розбити результат на частини та переглянути.

Приклад відповіді на успішний запит, що був описаний вище:


```
[
    {
   "additional_params": {
     "lang_id": 2,
     "page": 0,
     "view_type_id": 0,
     "target": "search",
…
   },
   "result": {
   "search_result": {
     "ids": [
        "26956399",
        "25855206",
     …
        "26874853"
     ],
     "count": 1,
     "last_id": 0,
     "qs": {
```

Дана відповідь API представлена у скороченій формі, оскільки повна версія складає понад 500 рядків і містить в собі перелік id оголошень, що задовольняють критерії, які були вказані у запиті.

Розглянемо клієнтський графічний інтерфейс AUTO.RIA. Він доступний за адресою https://auto.ria.com/ .

Рисунок 1.4. Головна веб-сторінка AUTO.RIA

На рисунку зображена головна сторінка сайту AUTO.RIA. Графічний інтерфейс є доволі простим та зрозумілим для користувача, при цьому має сучасний вигляд.

На головній сторінці виділяється 5 основних графічних елементів:

1. Вкладки, на яких є можливість вибрати ресурс компанії RIA.COM, що розміщує оголошення певної категорії: нерухомість, автотовари, запчастини чи автомобілі.

2. Форма пошуку оголошень за вказаними параметрами. Дані з цієї форми відправляються в точку прийому прикладного програмного інтерфейсу, що був продемонстрований вище. Тут звичайний користувач може вказати основні параметри для пошуку. Натиснувши кнопку «Пошук», буде отримано список оголошень за вказаними параметрами. Якщо параметри не вказані, користувач отримає всі оголошення, розміщені на сайті та відсортовані за датою, починаючи від найновіших.

3. Також даний віджет має кнопку «Розширений пошук», при натисненні на яку, користувач має можливість вказати більше параметрів, що зробить пошук більш детальним та зменшить діапазон знайдених оголошень.

4. Віджет під номером 3 складається з кнопок, що здійснюють навігацію по сайту AUTO.RIA. З їх допомогою можна перейти в розділ з новими авто, переглянути новини, що пов’язані з сферою автомобілів чи вибрати певні послуги, що надає сайт, наприклад: пошук станцій технічного обслуговування чи автомийок або підрахунок середньої ціни для вибраної моделі авто.

5. Кнопка для переходу в особистий кабінет користувача. Якщо користувач зареєстрований в системі AUTO.RIA, то він може вказати особисту інформацію про себе в кабінеті користувача, переглянути інформацію про розміщені ним оголошення, актуальні новини тощо.

6. Кнопка «Продати авто». Для розміщення оголошення не обов’язково потрібно пройти реєстрацію на AUTO.RIA. Натиснувши на кнопку, користувач перейде на сторінку створення оголошення, де він може заповнити основну інформацію про транспортний засіб, після чого розмістити оголошення. Для того, щоб оголошення було показане в пошуку, воно має пройти етап модерації, що може зайняти певний час.

Також на головній сторінці є віджет вибору мови. Доступні мови інтерфейсу: українська та російська. Нижче, під формою пошуку, розміщується стрічка, на якій відображаються нещодавно додані на сайт оголошення.

1.2.2 Avtobazar

Також вартий уваги ще один популярний веб-сайт з оголошеннями про продаж авто – avtobazar.ua. Avtobazar дещо відрізняється від AUTO.RIA, оскільки працює за принципом автомобільного ринку. Обмежень по типу транспорту тут немає, проте розділ зі спецтехнікою не представлений. Списки автосервісів відсутній, проте нещодавно було додано список салонів, де можна придбати ту чи іншу марку авто.

Значною перевагою Autobazar є те, що тут достатньо просто розмістити оголошення, необхідно заповнити лише 10 параметрів. Даний веб-сайт не надає свій API у відкритому доступі, з ним можливо взаємодіяти лише за допомогою графічного інтерфейсу користувача у веб-браузері.

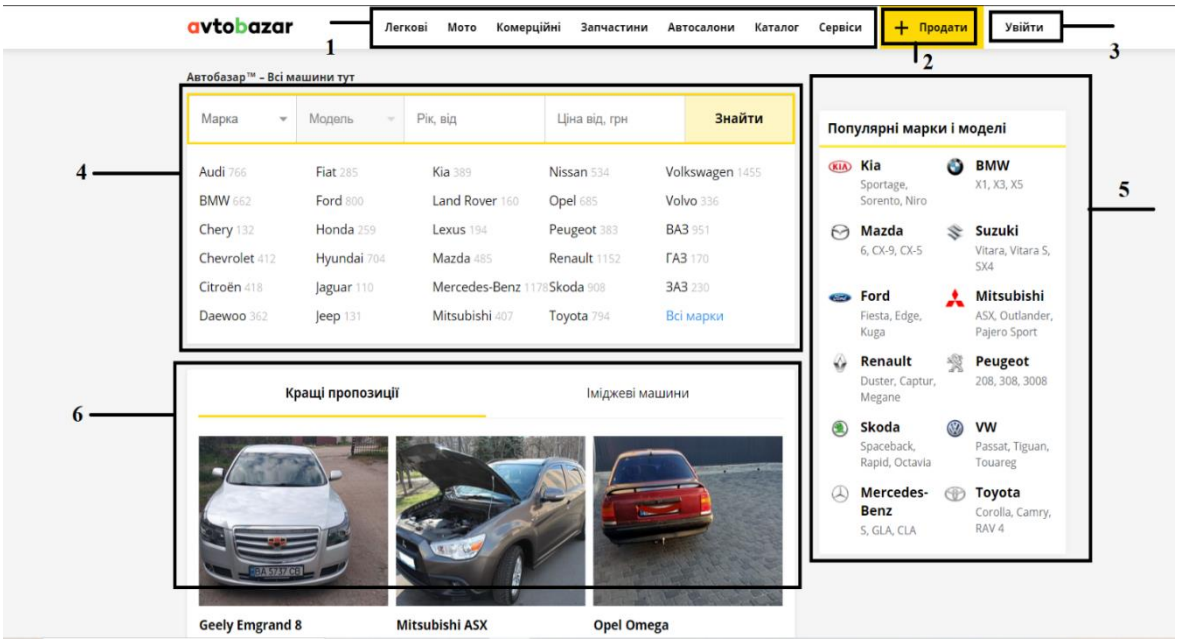


Рисунок 1.5 Головна веб-сторінка Avtobazar

Всі віджети, що відповідають за певний функціонал на головній сторінці Autobazar, розбиті на блоки, що виглядає одночасно доволі мінімалістично та зручно для користувачів. Виділяються основні 6 компонент, серед них:

1. Вкладки на верхівці сайту. Тут можна вибрати категорію транспорту або перейти до пошуку запчастин, автосалонів. Також у вкладці «Сервіси» доступний митний калькулятор та оцінка вартості авто. Оскільки верхівка завжди закріплена, це робить навігацію по сайту зручною і простою.
2. Кнопка «Продати» відкриває користувачу вікно створення нового оголошення, яке можна розмістити за декілька хвилин.

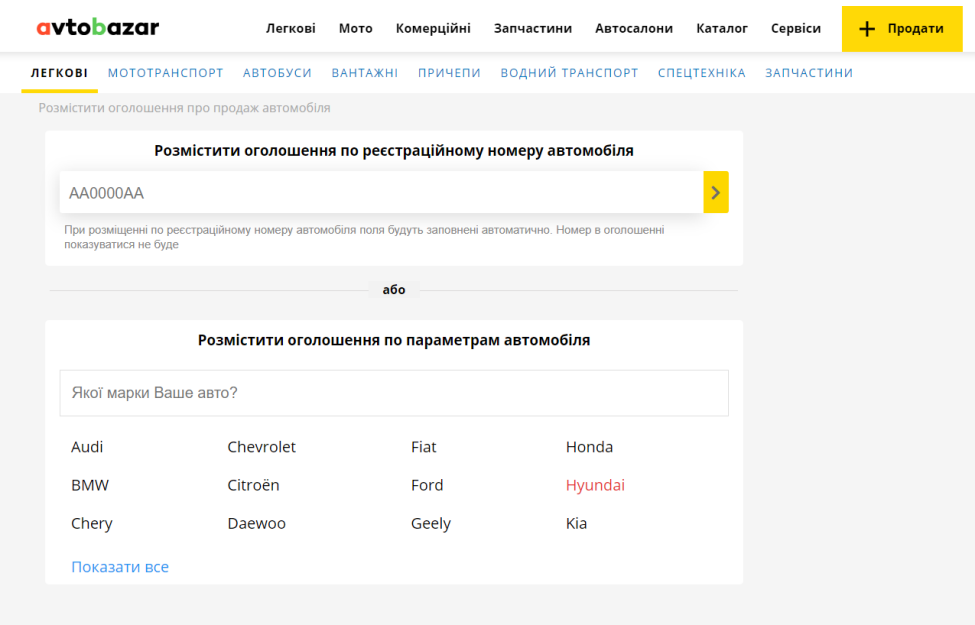


Рисунок 1.6. Меню створення оголошення на Avtobazar

3. Кнопка «Увійти» надає можливість авторизуватись на сайті і отримати доступ до особистого кабінету або пройти реєстрацію.
4. Форма пошуку оголошень. Тут можна вказати лише марку та модель, початкове значення року випуску та ціни. Вказати додаткові параметри можливо лише після вибору марки автомобіля, що є доволі незручно, якщо користувач бажає переглянути автомобілі від кількох виробників.
5. Таблиця «Популярні марки та моделі» дозволяє відслідковувати тенденції ринку. Вона відображає найпопулярніші марки виробників та їх моделі, які користувачі шукають на Avtobazar.
6. Блок вміщує в собі дві вкладки: «Кращі пропозиції» та «Іміджеві авто».

   «Кращі пропозиції» відображає найпопулярніші та найвигідніші пропозиції з точки зору ціна/рік випуску/комплектація автомобіля. «Іміджеві авто» надає інформацію про транспортні засоби бізнес- та преміум-класу, ціна на які перевищує мільйон гривень.

ВИСНОВОК ДО РОЗДІЛУ 1

В ході виконання першого розділу даної дипломної роботи було розглянуто декілька найпопулярніших платформ для продажу автомобілів в Україні. Проаналізувавши ряд таки систем, можемо зробити висновок, що лише AUTO.RIA надає свій прикладний програмний інтерфейс у відкритому доступі, однак для використання API без ліміту запитів до серверу вимагає від користувача оформлення комерційної співпраці. Також значним мінусом є те, що API надає лише доступ до загальної бази оголошень AUTO.RIA, що є доволі незручним для бізнесу, який планує придбати дану послугу, тому що немає можливості обмежити оголошення, розміщені на веб-сайті, що використовує прикладний програмний інтерфейс AUTO.RIA та між оголошеннями на самому AUTO.RIA. Також, варто зазначити, що AUTO.RIA це лише розділ з оголошеннями про автомобілі великої онлайн-дошки оголошень RIA.UA.

Avtobazar є самостійним ресурсом і повноцінною платформою, однак вони не надають жодного доступу безпосереднього до прикладного програмного інтерфейсу їх системи. Взаємодія можлива лише з допомогою клієнтського графічного інтерфейсу користувача у веб-браузері. Варто зазначити, що дизайн інтерфейсу є доволі мінімалістичним та сучасним. Також Avtobazar працює за концепцією автомобільного ринку, а не дошки оголошень. За останній рік розробники Avtobazar значно покращили функціонал сайту. Було додано такі можливості, як:

- Список автосалонів;
- Калькулятор розмитнення автомобілів;
- Калькулятор оцінки вживаного авто.

РОЗДІЛ 2
ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Система, що представлена в даній дипломній роботі розроблена з використанням мови програмування Java та фреймворку Spring Boot, що використовується зокрема для створення серверної частини веб-додатків та прикладних програмних інтерфейсів. Для створення демонстраційного графічного інтерфейсу користувача застосовано фреймворк Angular з використанням мови програмування TypeScript, мови розмітки гіпертексту HTML та каскадних таблиць стилей CSS. Всі дані, що потрібні для коректної роботи системи, зберігаються в документо-орієнтованій базі даних MongoDB.

2.1 Мова програмування Java

Java – об’єктно-орієнтована мова програмування, що була створена американською компанією «Sun Microsystems» у 1995 році, як основний компонент платформи Java. У 2009 році «Sun Microsystems» була придбана корпорацією «Oracle», тому потенційним розвитком Java почали займатись саме «Oracle».[2]

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. [3]

Передусім Java розроблялась як платформо-незалежна мова програмування, що було досягнуто компіляцію програмного коду в байт-код, який інтерпретується віртуальною машиною на тій чи іншій платформі. «Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Через те, що програмне забезпечення створена на Java не залежить від платформи, мова має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що у порівнянні, наприклад, з C++ зменшує швидкість роботи програмного додатку. Однак, за необхідності більш низькорівневих операцій, ніж ті, що надає Java, є можливість викликати підпрограми, написані іншими мовами, використовуючи JNI.[4]

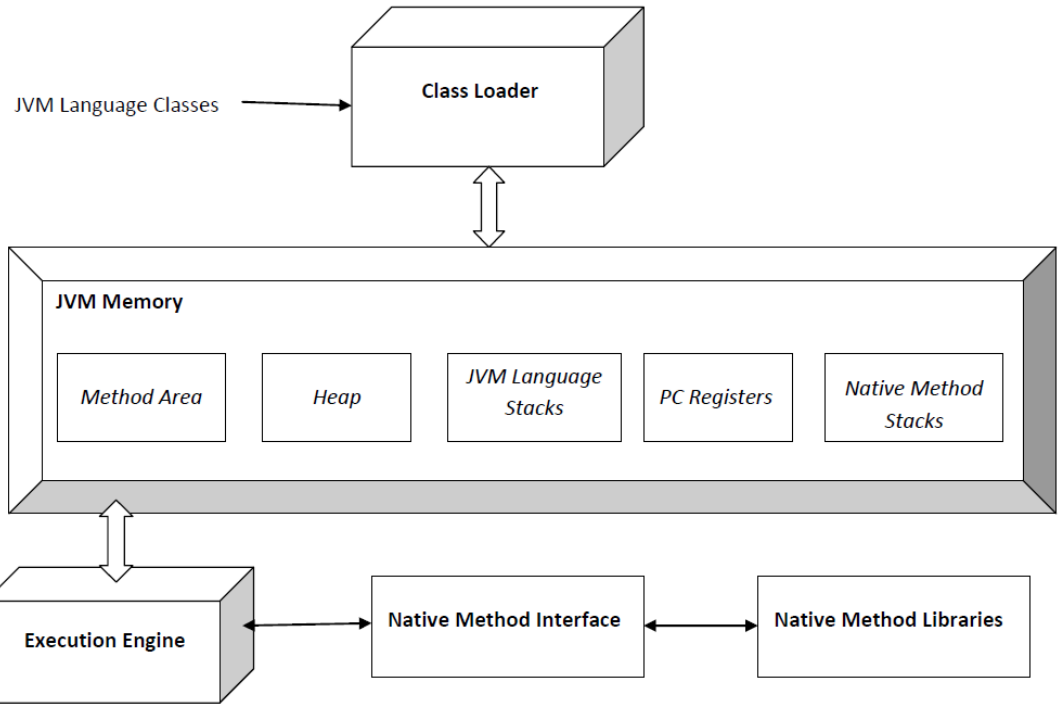


Рисунок 2.1. Схема роботи віртуальної машини Java

Java швидко знайшла популярність у мільйонів інженерів у всьому світі через свою платформо-незалежну архітектуру, об'єктно-орієнтованість та автоматизацію роботи з пам'яттю. З 1995 року було написано безліч проєктів різного рівня з використанням Java та поріднених з нею технологій. Сьогодні Java частіше всього використовується для створення клієнт-серверного програмного забезпечення, а саме для розробки серверної частини додатків.

Java займала перше місце у авторитетному рейтингу мов програмування TIOBE, починаючи з 2001 році. Лише у травні 2020 року Java поступилась мові програмування C, зайнявши другу сходинку. Двічі: у 2005 та 2015 роках, TIOBE визнавали Java мовою програмування року в світі.

Код написаний на Java компілюється в байт-код, який представляє собою спрощений машинний код. Байт-код можна запустити на будь-якій платформі, що має встановлену віртуальну машину JVM. JVM інтерпретує байт-код у машинний код пристосований до специфіки певної операційної системи та процесора. Стандартні бібліотеки Java, що входять до JDK, надають високорівневий доступ до таких залежних від платформи особливостей, як графіка та її елементи, потоки та процеси, робота з мережами.[4]

Основною перевагою байт-коду є портативність, однак програми написані на Java працюють повільніше, ніж програми написані на мовах програмування, що зразу компілюються у машинний код. Причиною цьому є додаткові витрати ресурсів на інтерпретацію. Проте, сучасні реалізації віртуальної машини Java суттєво скоротили розрив у швидкодії, запровадивши певні методи оптимізації.

Одним із таких методів є JIT-компілятор, який перетворює байт-код Java у машинний код під час першого виконання програми, а потім зберігає його у кеш. Така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання. Складніші віртуальні машини також використовують динамічну рекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково рекомпілює та оптимізує певні її частини. З використанням динамічної рекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про довкілля періоду виконання та про завантажені класи. До того ж він може виявляти так звані гарячі точки (англ. *hot spots*) — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна рекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.[5]

Ще один спосіб оптимізації байт-коду, широко відомий як статична компіляція, або компіляція *ahead-of-time* (AOT). Цей метод передбачає, як і традиційні компілятори, безпосередню компіляцію у машинний код. Це забезпечує хороші показники в порівнянні з інтерпретацією, але за рахунок втрати незалежності від платформи: програму, що була скомпільована AOT-компілятором можна запустити лише на певній платформі.[6]

Java є більш об'єктно-орієнтованою, ніж C++, оскільки всі її дані та дії групуються в класи об'єктів. Java не вважається повністю об'єктно-орієнтованою мовою через те, що в ній присутні примітивні типи (int, float, double, char, byte, short, long, boolean). Всі об'єкти в Java наслідують об'єкт Object, який знаходиться на вершині дерева наслідування, з нього вони успадковують базову поведінку об'єкта.

Для автоматичного керування пам'яттю, віртуальна машина Java використовує збирач сміття Garbage Collector. Всі об'єкти під час виконання програми зберігаються в купі (англ. Heap). Розробник вирішує на якому етапі створювати об'єкт, а збирач сміття відповідає за те, щоб його видалити. Об'єкт вважаться непотрібним і видаляється з купи коли на нього немає посилань.

2.2 Фреймворк Spring

Spring Framework – універсальний фреймворк з відкритим програмним кодом для Java-платформи.

Фреймворк був вперше випущений під ліцензією Apache 2.0 в червні 2003 року. Перша стабільна версія 1.0 була випущена в березні 2004. Spring 2.0 був випущений в жовтні 2006, Spring 2.5 - в листопаді 2007, Spring 3.0 в грудні 2009, і Spring 3.1 в грудні 2011. Поточна версія - 5.2.4.

Незважаючи на те, що Spring не забезпечував якусь конкретну модель програмування, він став широко поширеним в сфері споріднених з Java технологій, головним чином як альтернатива і заміна моделі Enterprise JavaBeans.

Spring надає велику свободу Java-розробникам в проектуванні; крім того, він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні додатків корпоративного масштабу.

Особливості ядра Spring можуть бути застосовані в будь-якому Java-додатку, також існує певний ряд розширень і удосконалень для побудови веб-додатків на платформі Java EE. Саме тому Spring набув великої популярності і визнається розробниками як стратегічно важливий фреймворк.[7]

Spring забезпечує вирішення багатьох проблем, з якими стикаються Java-розробники та компанії, які хочуть створити програмну систему, під управлінням платформи Java. Через широку функціональність важко визначити найбільш важливі модулі, оскільки кожен з них відповідає за вирішення проблем певної сфери. Spring не повністю пов'язаний з платформою Java Enterprise, незважаючи на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring відомий як набір розширень, потрібних для ефективної розробки складних бізнес-додатків. Ще одна його перевага в тому, що він запровадив раніше непопулярні функціональні можливості в домінуючі методи розробки сьогодення, навіть поза платформою Java.

Spring може бути розглянутий розробником як колекція спеціалізованих фреймворків. Більшість цих модулів може працювати незалежно один від одного. Вони діляться на структурні елементи типових комплексних програм:

- IoC-контейнер - конфігурує компоненти програмних додатків та керує життєвим циклом Java-об'єктів.
- Фреймворк аспектно-орієнтованого програмування - працює з функціоналом, що не може бути реалізований з допомогою підходів об'єктно-орієнтованого програмування на Java без суттєвих проблем.
- Фреймворк роботи з даними - працює з системами керування базами даних, використовуючи JDBC- і ORM-засоби і забезпечує доступ до баз даних через прикладний програмний інтерфейс Java.

- Фреймворк керування транзакціями - координує між собою API керування транзакціями і засоби управління транзакціями для об'єктів Java.
- Фреймворк MVC - програмний каркас, заснований сервлетах з використанням HTTP протоколу, що надає змогу швидко будувати веб-додатки.
- Фреймворк віддаленого доступу - керує передачею Java-об'єктів через мережу в стилі віддаленого виклику процедур - RPC, підтримуючи технології RMI, CORBA, протоколи, що базуються на HTTP, включаючи web-сервіси (SOAP).
- Фреймворк аутентифікації і авторизації – дає змогу налаштувати засоби аутентифікації та авторизації для додатку.
- Фреймворк віддаленого управління - конфігурує управління Java-об'єктами для локальної або віддаленої конфігурації за допомогою JMX.
- Фреймворк роботи з повідомленнями - налаштовує реєстрацію об'єктів-слухачів повідомлень для обробки повідомлень з черги за допомогою JMS.
- Фреймворк для тестування - програмний каркас, що підтримує класи для написання модульних і інтеграційних тестів з використанням Junit та Mockito.

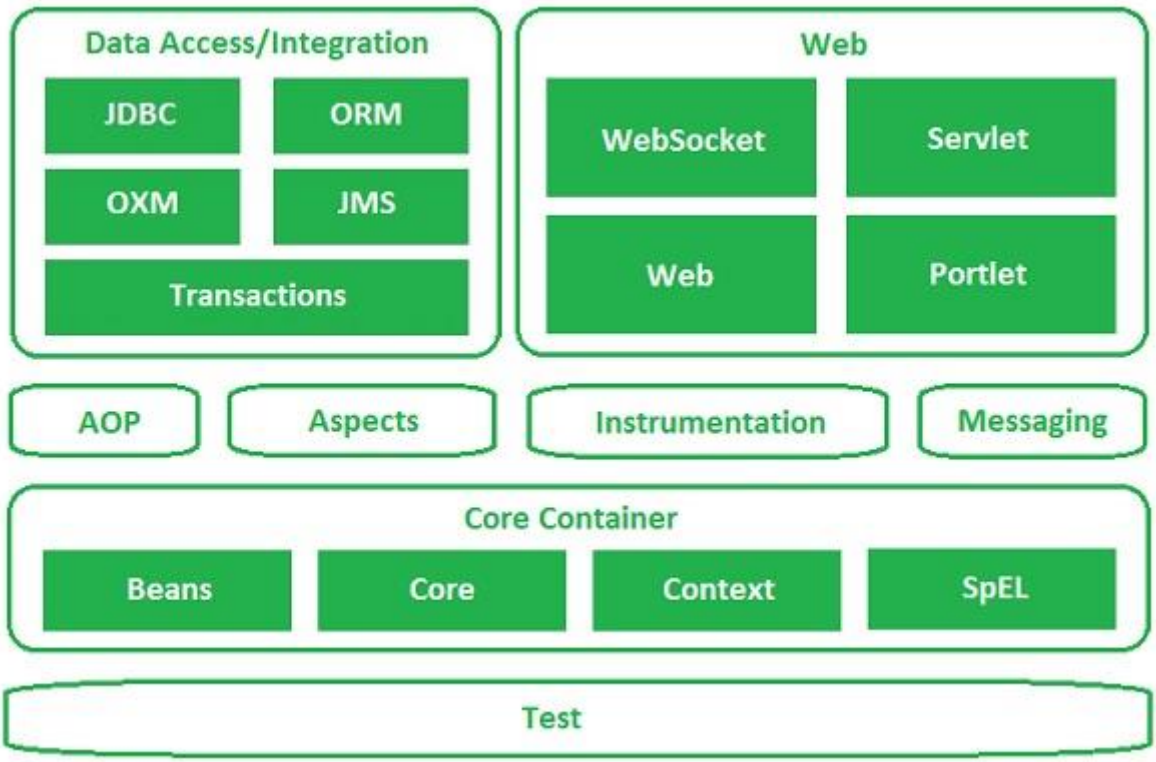


Рисунок 2.2. Схема компонентів Spring Framework

Центральною частиною Spring є контейнер Inversion of Control, який надає засоби налаштування та керування об'єктами Java за допомогою механізму рефлексії. Контейнер відповідає за управління життєвим циклом об'єкта: створення об'єктів, виклик методів ініціалізації, а також їх налаштування, шляхом зв'язування об’єктів між собою.

Об'єкти, що створюються контейнером, також називаються керованими об'єктами або бінами (англ.beans). Зазвичай, конфігурація контейнера, здійснюється шляхом впровадження анотацій, але є можливість, завантажити XML-файли, що містять визначення бінів і надають інформацію, необхідну для їх створення.[8]

Об'єкти можуть бути отримані одним з двох способів:

- Пошук залежності - шаблон проєктування, в якому об'єкт запитує у контейнера екземпляр об'єкта з певним ім'ям або певного типу.

- Впровадження залежності - шаблон проєктування, в якому контейнер передає екземпляри об'єктів по їх імені іншим об'єктам за допомогою конструктора, поля об’єкта або метода встановлення поля.

Spring має власний MVC-фреймворк для побудови веб-додатків. Клас DispatcherServlet є основним контролером Spring MVC і відповідає за делегування управління різним інтерфейсам, на всіх етапах виконання HTTP-запиту.

Spring MVC надає розробнику наступні можливості:

- Ясний і прозорий поділ між рівнями в MVC та в запитах.
- Кожен інтерфейс робить тільки свою частину роботи.
- Інтерфейс завжди може бути замінений альтернативною реалізацією.
- Інтерфейси тісно пов'язані з Servlet API.
- Високий рівень абстракції для веб-додатків.

Spring надає свій інтерфейс для доступу до баз даних за допомогою JDBC. Крім того, він підтримує всі популярні ORM: Hibernate, JPA, JDO, EclipseLink, iBatis, Apache OJB, Apache Cayenne.

Для всіх цих фреймворків, Spring надає такі особливості:

- Автоматичне отримання і звільнення ресурсів бази даних.
- Обробка виключних ситуацій.
- Підтримка транзакції в операціях з даними.
- Отримання об'єктів бази даних з пулу з'єднань.
- Абстракція для обробки BLOB і CLOB.

Фреймворк управління транзакціями в Spring надає механізм абстракцій для платформи Java. Основні можливості цих абстракцій:

- Робота з локальними і глобальними транзакціями.
- Робота з вкладеними транзакціями.
- Робота з точками збереження в транзакціях.

2.3 Система керування базами даних MongoDB

MongoDB – документо-орієнтована система керування базами даних з відкритим програмним кодом.

На відміну від реляційних систем керування базами даних, MongoDB не потребує опису таблиць та встановлення зв’язків між ними.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційних баз даних.

З початком ери розвитку сфери інформаційних технологій та інтенсивного розвитку розробки різних програмних продуктів стало звичним зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних чи ні.[9]

Завдяки документо-орієнтованій моделі, MongoDB працює швидше, має кращу масштабованість, її легше використовувати під час розробки програмного забезпечення. Але, навіть враховуючи всі недоліки традиційних баз даних і переваги MongoDB, важливо розуміти, що завдання та методи їх вирішення суттєво різняться між собою. В певній ситуації MongoDB дійсно поліпшить продуктивність програми, наприклад, якщо система потребує збереження складної структури даних. В іншому випадку оптимальніше використовувати реляційні бази даних. Зокрема, можна використовувати змішаний підхід збереження даних: зберігати один тип даних в документо-орієнтованій MongoDB, а інший тип даних – в реляційній базі даних.

Базу даних під управлінням СКБД MongoDB можна розгорнути на декількох фізичних серверах. MongoDB дозволяє легко обмінюватись даними між серверами та зберігати їх цілісність.

MongoDB реалізована з використанням мови програмування C++, що дозволяє встановлювати її на різних платформах. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Її програмний код поширюється під ліцензією AGPLv3.

Одним з популярних стандартів обміну та збереження даних є JSON. JSON ефективно описує складні за структурою дані.

Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON, що представляє з себе бінарний формат JSON.

BSON дозволяє працювати з даними швидше, ніж з JSON: швидше виконується пошук і обробка інформації в базі даних. Порівняно з JSON, дані у форматі BSON займають більше пам'яті на пристрої зберігання, проте це компенсується швидкістю взаємодії з даними.

В реляційних базах даних інформація зберігається в рядках, що формують собою таблиці. В MongoDB дані зберігаються як документи. Документи можуть зберігати складні структури даних, що є вагомою перевагою над рядками реляційних баз даних, що можуть зберігати лише певні типи у певному стовпці. Документ у MongoDB представляє собою набір ключів, кожному з яких відповідає певне значення. Ключ – це символьна мітка, з якою пов'язаний визначений сегмент даних документа.

В MongoDB для кожного документа присвоюється унікальний ідентифікатор, який називається _id. Якщо при створенні документа не вказати його явно, то MongoDB згенерує його автоматично. Дане поле повинно бути унікальним в межах однієї колекції. При спробі додати два документи з однаковим ідентифікатором, буде доданий лише один з них, а при спробі додати інший, буде отримана помилка. Якщо ідентифікатор явно не вказаний розробником, то MongoDB згенерує спеціальне значення розміром 12 байт. Це значення складається з декількох частин:

- значення типу timestamp, розміром в 4 байти, що містить в собі дату і час на конкретний момент;
- ідентифікатор обчислювальної машини, на якій працює MongoDB, розміром в 3 байти;
- ідентифікатор процесу – 2 байти;
- лічильник – 3 байти.

Таким чином, перші 9 байт ідентифікатора гарантують унікальність документа серед інших обчислювальних машин, на яких можуть бути розміщені копії бази даних. Останні 3 байти гарантують унікальність протягом однієї секунди для одного процесу. Така модель генерації унікального ідентифікатора є ефективною, оскільки дозволяє гарантувати максимально можливу унікальність і можливість створення 16 777 216 унікальних ідентифікаторів на секунду в одному процесі.

На відміну від реляційних баз даних, де при відсутності значення для певного поля, в рядок записується значення NULL, в MongoDB ключ, якому не відповідає жодне значення, ігнорується і в документі ніяк не фігурує.

Аналогією таблицям, що характерні реляційним системам керування базами даних в MongoDB є колекції. В реляційних базах даних таблиці зберігають жорстко структуровані однотипні об’єкти, проте колекції MongoDB можуть зберігати різноманітні за структурою та набором властивостей дані.

Система зберігання даних в MongoDB представляє собою набір реплік. Він містить основний вузол, а також може мати в собі певну кількість побічних вузлів. Всі побічні вузли зберігають цілісність і автоматично оновлюються разом з головним вузлом репліки. Якщо основний вузол з тих чи інших причин виходить з ладу, то один з побічних вузлів стає головним.

Відсутність конкретної схеми бази даних, як в реляційних БД є значною перевагою MongoDB. В зв'язку з цим немає потреби при зміні концепції чи структури зберігання даних створювати схему знову, що значно полегшує роботу з базою даних та її можливим подальшим масштабуванням. Крім того, це дозволяє значно зекономити час розробників програмного забезпечення. У них більше немає потреби створювати заново структуру бази даних і витрачати час на побудову складних запитів для маніпуляцій з даними.

Значною проблемою при роботі з будь якою системою керування базами даних є збереження даних великого розміру.

Деякі СУБД реалізують певний спеціальний тип для таких випадків, щоб зберігати бінарні дані в базі даних, як, наприклад, тип BLOB в системі управління БД MySQL.

На відміну від реляційних баз даних, MongoDB дозволяє зберігати документи з різним набором даних, проте розмір цих документів обмежується 16 мегабайтами. При потребі зберігати дані більшого розміру, MongoDB пропонує спеціальну технологію GridFS, що дозволяє зберігати дані, розмір яких перевищує 16 мегабайт. Ця система складається з двох колекцій, що пов'язані між собою. Перша колекція, що називається files, зберігає імена файлів, а також їх метадані, такі як розмір, розширення тощо. В другій колекції, якій присвоєна назва chunks, зберігаються дані файлів, розбиті на сегменти розміром 256 кілобайт. Пакет mongodb має спеціальну утиліту mongofiles, що дозволяє провести тест системи GridFS.

Виділимо основні можливості MongoDB:

- Документо-орієнтоване модель збереження даних у форматі BSON
- Гнучка мова для створення запитів
- Динамічні запити
- Повна підтримка індексів
- Профілювання запитів
- Швидке оновлення даних після їх зміни
- Ефективне зберігання бінарних даних великих обсягів
- Запис до журналу операцій, що модифікують дані в базі
- Стійкість до відмов та простота масштабованості
- Може працювати відповідно до парадигми MapReduce

Для встановлення MongoDB потрібно завантажити один з дистрибутивів з офіційного сайту розробників.

Офіційний сайт пропонує дистрибутиви для різних операційних систем, зокрема: Windows, Linux, MacOS, Solaris. Є два види системи управління базою даних MongoDB:

- Community – для некомерційного користування, містить в собі стандартний набір засобів та утиліт;
- Enterprise – комерційна версія, має більше можливостей та використовується компаніями, що розроблюють програмне забезпечення.

На момент написання даної бакалаврської роботи останньою версією MongoDB є реліз 4.2, який і був використаний в роботі.

Після встановлення MongoDB, користувачу доступний перелік утиліт, що виконують певну роль при розробці програмного продукту:

- bsondump – читає контент BSON-файлів і перетворює їх в зручний для відображення формат, наприклад, в JSON;
- mongo - консольний інтерфейс для взаємодії з базами даних та керування ними;
- mongod - сервер баз даних MongoDB. Сервер обробляє запити, керує даними і виконує різні операції в фоновому режимі, що стосуються управління базами даних;
- mongodump - утиліта для створення резервних копій інформації з баз даних;
- mongoexport - утиліта для трансфрмації даних в формати JSON, TSV або CSV;
- mongofiles - утиліта, що дозволяє керувати файлами в системі GridFS;
- mongoimport - утиліта, що імпортує данні у форматах JSON, TSV або CSV;
- mongorestore - дозволяє зчитувати дані з буфера, що був створений mongodump та записувати в нову або існуючу базу даних;
- mongos - служба маршрутизації MongoDB, яка допомагає обробляти запити і визначати місце розташування даних в кластері MongoDB;

- mongorestat – відповідає за лічильники виконаних операцій з базою даних;
- mongotop - надає можливість підрахувати час, що був витрачений на операції читання даних та їх запису в базі даних.

Також, для більш зручної роботи з MongoDB, замість консольного інтерфейсу можна використовувати графічний клієнт, що має назву Compass. Його також можна завантажити з офіційного веб-сайту.

Після підключення до серверу MongoDB, перед користувачем з'явиться список баз даних, що присутні на даному сервері.

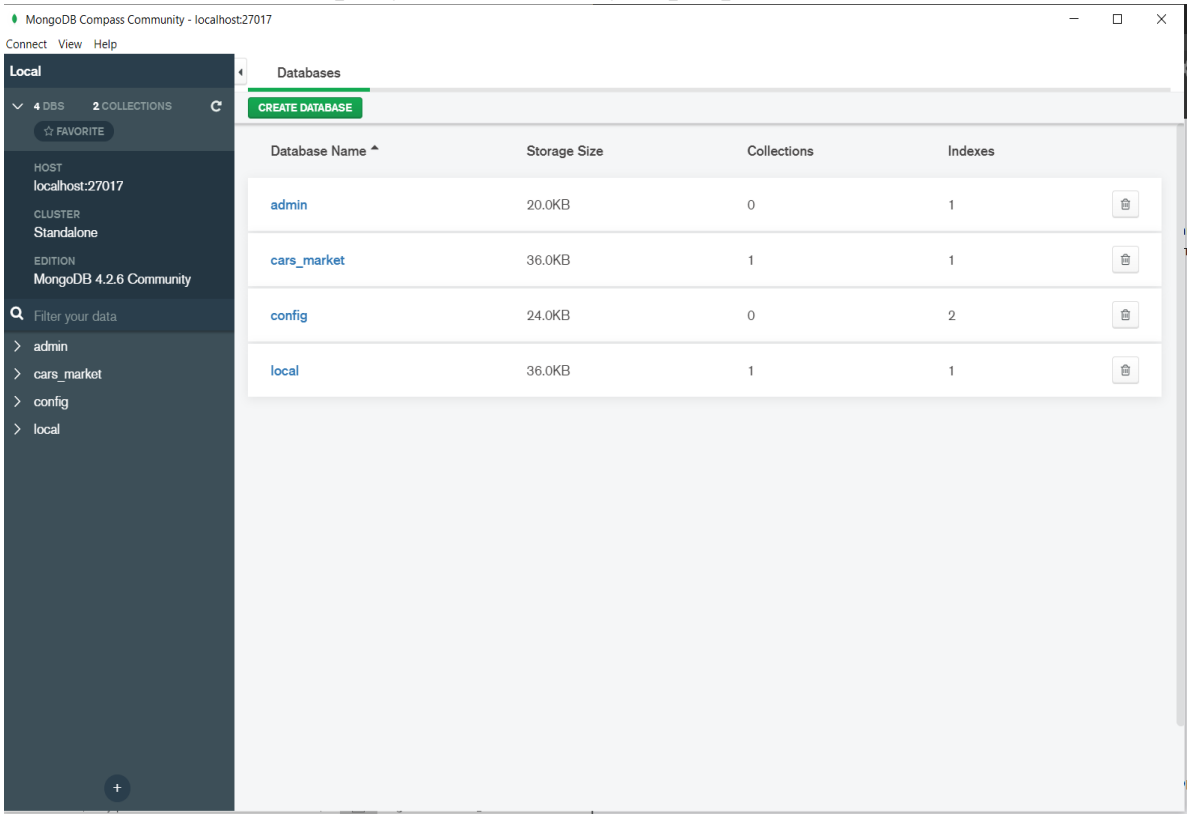


Рисунок 2.3. Список баз даних MongoDB в графічному клієнті Compass

Після вибору певної бази даних користувач має можливість отримати інформацію про неї. Наприклад, набір колекцій у вибраній базі даних та розмір пам'яті, що вона займає на сервері.

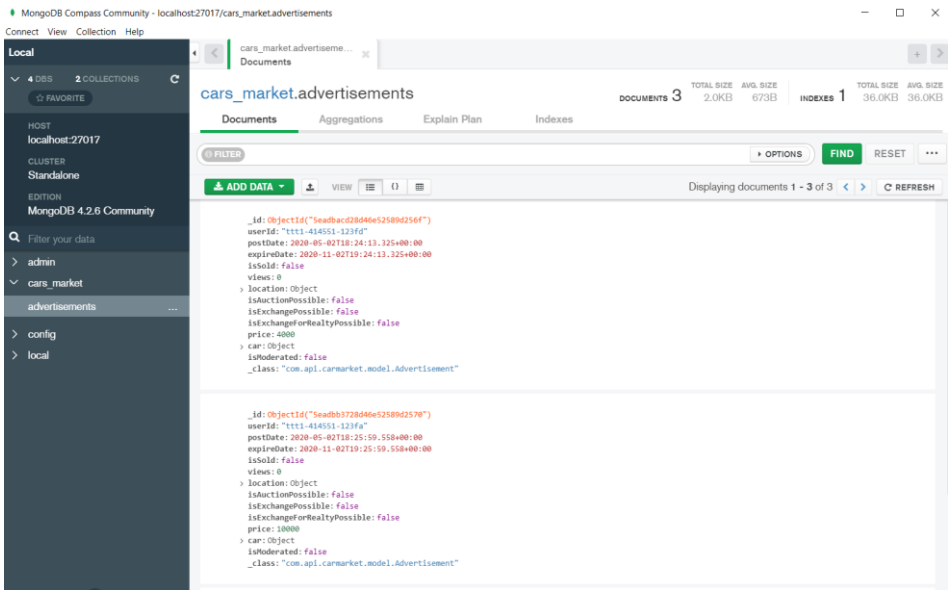


Рисунок 2.4. Графічне відображення даних колекції в базі даних MongoDB

2.4 Веб-фреймворк Angular

Angular (іноді Angular 2+, що означає версію 2 та вище) – фреймворк для створення клієнтського веб-інтерфейсу з відкритим програмним кодом. Angular розробляється групою розробників Angular Team, що входить до складу корпорації Google. Фреймворк написаний на мові програмування TypeScript.

Перед усім Angular застосовується для створення SPA-додатків, тобто клієнтських додатків, що складаються з однієї сторінки. В цьому контексті, Angular унаслідує інший фреймворк, що має назву AngularJS. Проте, Angular, це не більш нова версія AngularJS, а повністю новий фреймворк, написаний з використанням TypeScript. AngularJS – це фреймворк для мови програмування JavaScript.

Angular надає таку функціональні можливості, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі.[9]

Angular підтримує мови програмування Dart та JavaScript, що дає можливість використовувати їх в створенні проєктів, проте основною мовою програмування фреймворку все ж таки вважається TypeScript.

Фреймворк Angular користується популярністю серед розробників завдяки можливості розбивати програмний код проєкту на модулі, які пізніше можна

використати знову в інших проєктах. Кожен додаток на Angular складається з кількох окремих модулів. Також, кожен додаток має принаймні один головний модуль (root module), який за замовчуванням прийнято називати AppModule.

Одним з ключових елементів додатку створеного з допомогою Angular є компонент. Компонент керує відображенням даних на екрані користувача. Ще однією важливою складовою додатку на Angular є сервіс. Сервіси в Angular представляють собою доволі широкий набір класів, які виконують певні специфічні задачі.

Наприклад, роботу з даними, обчислення, логування, конвертацію сутностей тощо. На відміну від компонент, сервіси ніяк не впливають на відображення даних, вони керують розміткою HTML і не впливають на неї безпосередньо. Сервіси виконують чітко визначену задачу.

Стандартні завдання сервісів:

- Надавати дані для додатку. Сервіс може сам зберігати дані в пам'яті, або для звертатися отримання даних до будь-якого джерела даних, наприклад, до сервера.
- Сервіс може представляти канал взаємодії між окремими компонентами програми.
- Сервіс може інкапсулювати бізнес-логіку, різні обчислювальні завдання, завдання по логуванню подій, які краще виносити з компонентів. Тим самим код компонентів буде зосереджений безпосередньо на роботі з відображенням. Крім того, тим самим ми також можемо вирішити проблему повторення коду, якщо нам буде потрібно виконати одну і ту ж задачу в різних компонентах і класах додатку.

2.5 Мова програмування TypeScript

TypeScript – мова програмування, що була розроблена корпорацією Microsoft та презентована восени 2012 року. Насамперед, TypeScript позиціонує себе як мова для розробки веб-додатків, що розширює можливості JavaScript.

TypeScript є зворотно сумісним з JavaScript, оскільки програмний код, що був написаний на TypeScript після компіляції перетворюється в код на JavaScript. Фактично, після компіляції додаток створений з використанням TypeScript можна запустити в будь-якому браузері, або в системі, що використовує серверну платформу Node.js. В програмному коді TypeScript є можливість викликати функції стандартної бібліотеки JavaScript.

Також можливо створювати свої функції та програмні модулі, які можна вільно використовувати в програмі на TypeScript.

Варто виділити основні переваги TypeScript над JavaScript:

- Можливість явного присвоєння типу змінній (статична типізація).
- Підтримка створення та використання класів (як в об'єктно-орієнтованих мовах програмування).
- Підтримка модульності додатку.
- Підтримка концепцій ООП.
- Простота переходу розробників з інших строго типізованих мов програмування.

Також, варто зазначити, що на відміну від JavaScript, TypeScript є мовою, що компілюється. Компілятор успішно зберігає принцип динамічної типізації JavaScript, проте він дозволяє виявити помилки, ще на етапі компіляції, що є неможливим при використанні класичного JavaScript.

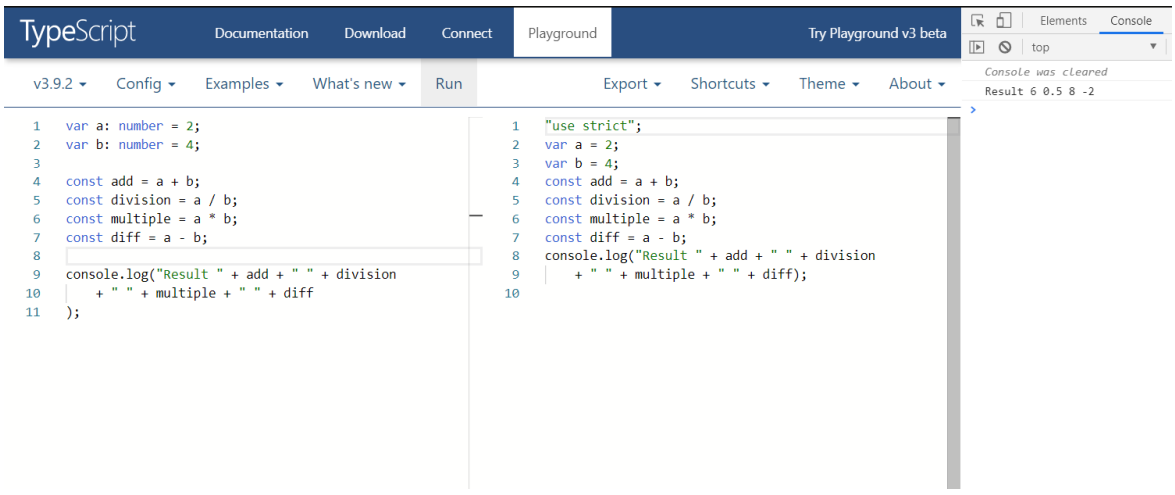


Рисунок 2.5. Приклад програми на TypeScript

2.6 Механізм впровадження залежностей

Механізм впровадження залежностей притаманний більшості сучасних фреймворків, в тому числі Angular 2+ та Spring. Він дозволяє встановлювати між програмними компонентами так званий «слабкий зв’язок», що робить систему більш універсальною і здатною масштабуватись.

Завдяки впровадженню залежностей заміна програмних компонентів продукту є швидкою і простою. Даний механізм є реалізацією шаблону проєктування «Впровадження залежностей».

Впровадження залежності (англ. Dependency Injection) – шаблон проєктування програмного забезпечення, що відноситься до типу породжуючих (твірних) шаблонів. Надає зовнішню незалежність програмному компоненту, використовуючи принцип інверсії управління для того, щоб отримати та впровадити залежності.

Впровадження – це процес передачі сервісу (залежності) клієнту (залежному об’єкту). Передача залежності клієнту замість надання йому можливості створити сервіс є фундаментальною вимогою шаблону проєктування.

Існує три загальноприйняті форми впровадження залежностей:

- Через конструктор класу.
- Через поле класу.
- Впровадження через метод класу, що встановлює значення поля.

2.7 Мова розмітки HTML

HTML – це мова розмітки гіпертексту, яка переважно використовується для створення документів в мережі Інтернет. HTML було створено на основі мови SGML, що пояснює концепцію визначення типу документу та структурної розмітки тексту. Основною структурою в HTML є тег. Кожен тег описує певний елемент HTML-сторінки. Веб-браузери виконують парсинг HTML-документа і будують DOM-дерево, згідно вказаним в документі тегам. Після інтерпретації, користувач отримає відображення даних на екрані пристрою.

В 2014 році була завершена робота над новим стандартом HTML5, який приніс багато нововведень у сферу веб-розробки.

Зокрема:

- Новий алгоритм парсингу для побудови DOM-дерева.
- Нові теги, наприклад, video та audio, які допомагають розмістити медіа контент на сторінці.
- Перевизначення правил та семантики елементів HTML, що вже існували.

З виходом нового стандарту, HTML перейшов на новий рівень, область його використання вийшла за межі веб-розробки: HTML5 використовується для створення мобільних додатків для смартфонів, що працюють на базі Android, iOS, Windows Mobile, а також, частково, для створення додатків для персональних комп'ютерів на базі операційної системи Window 8 і новіше.

HTML елементи утворюють структурні блоки сторінки для зображень, тексту, відео чи інтерактивних форм з різноманітним вмістом.

Дана мова розмітки дозволяє створювати структуровані сторінки за допомогою семантичного розподілу тексту на заголовки, параграфи, посилання, переліки, цитати тощо. Структурні елементи формуються за допомогою використання тегів (рисунок 2.6).

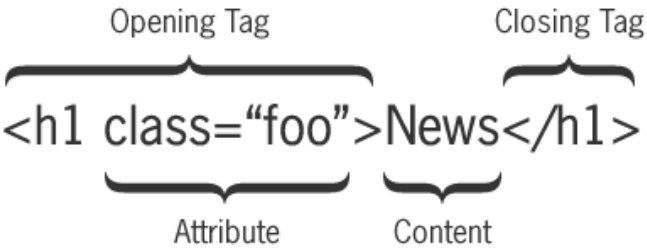


Рисунок 2.6. Структура тегів HTML

2.8 Каскадні таблиці стилей CSS.

CSS – спеціальна мова, що використовується для опису стилю сторінок, що були створенні за допомогою мов розміток даних. Зазвичай використовується для встановлення стилів HTML-сторінок, проте може бути використана з іншими мовами розмітки, наприклад, XML.

У сучасній веб-розробці каскадні таблиці стилей – це безальтернативна технологія для візуального оформлення веб-сторінок або користувацьких графічних інтерфейсів додатків.

Основною задачею CSS є відокремлення змісту сторінки та його графічної презентації: шрифти тексту, відступи від елементів сторінки та їх кольори тощо.

Правило CSS (рисунок 2.7) складається з двох основних частин: селектор, який вибирає ті структурні частини документу, до яких правило застосовується і блок декларації, де оголошують значення властивостей.

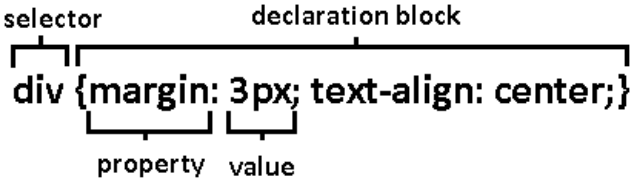


Рисунок 2.7. Структура правила в CSS

Існує декілька видів селекторів, серед яких найбільш популярними є: універсальний селектор, селектор класів, селектор ідентифікаторів, селектор елементів, селектор нащадків та селектор дочірніх елементів.

Головна відмінність між класом та ідентифікатором в тому, що окремий клас може бути присвоєний декільком елементам в межах документу, а ідентифікатор – унікальне значення в межах документу, що присвоюється тільки одному елементу. Також відмінність у тому , що можуть існувати множинні класи, коли клас окремого елемента складається з декількох слів, розділених пробілами. У такому випадку до елемента будуть застосовані правила для усіх класів у множині. Для ідентифікаторів таке неможливо

ВИСНОВОК ДО РОЗДІЛУ 2

В другому розділі даної бакалаврської роботи був описаний перелік основних технологій, що будуть використані під час розробки прикладного програмного інтерфейсу для системи продажу автомобілів. Було розглянуто стек технологій як для серверної частини продуту, так і для клієнтської.

Для створення прикладного програмного інтерфейсу буде використано мову програмування Java разом з фреймворком Spring, що є найпопулярнішою комбінацією технологій в сучасній розробці комерційних програмних продуктів.

Для демонстраційного клієнтського інтерфейсу буде використано веб-фреймворк Angular та мова програмування TypeScript.

Всі використані технології були детально описані та розглянуті, з вказаними перевагами та недоліками. Детально опрацьовано документо-орієнтовану СУБД MongoDB, оскільки правильна робота системи значно залежить від вибору бази даних та її функціонування.

РОЗДІЛ 3
ОПИС ПРИКЛАДНОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ

3.1 Опис основних операцій

Основні функції системи, що розробляється в рамках даної дипломної роботи базуються на чотирьох функціях управління даними. Вони також відомі як CRUD-операції, що є абревіатурою від англійських слів: create (створювати), read (читати), update (оновити) та delete (видалити).

В стандартних випадках, прикладний програмний інтерфейс має надавати можливість виконувати операції, що були описані вище, над усіма сутностями, які описують певний об’єкт з реального світу в рамках системи, що розробляється.

Оскільки, всі сутності, що використовуються в системі, зберігаються в базі даних, то CRUD-операції виконуються над записами у базі даних з використанням спеціалізованих програмних інтерфейсів, що дозволяють керувати даними, використовуючи певну мову програмування.

Функції управління даними можуть розглядатись не лише в рамках операцій над даними, що зберігаються в базі даних, а також стосовно інтерфейсу користувача. Наприклад, в програмному інтерфейсі для системи продажу автомобілів базовою сутністю є оголошення про продаж транспортного засобу. Тому, програмний інтерфейс повинен надавати користувачеві мінімальний набір функцій, зокрема:

- Створення (додавання) нових оголошень.
- Пошук оголошення за унікальним ідентифікатором.
- Редагування параметрів оголошення.
- Видалення існуючого оголошення з системи.

Також, варто виділити ще одну операцію, котра не входить в перелік CRUD-операцій, проте є фундаментальною для реалізації прикладного програмного інтерфейсу.

Операція має назву List (англ. записувати, створювати список) – відповідає за пошук в системі всіх об’єктів, яким властиві значення параметрів, що були вказані в пошуку. Описана функція програмного інтерфейсу виконує загальний пошук по системі. Наприклад, пошук всіх оголошень, що були розміщені сьогодні і у яких місто автора оголошення Київ.

3.2 Опис моделі даних системи

Модель даних – абстрактне представлення об’єктів реального світу, що мають відношення до програмної системи і безпосередньо використовуються в її рамках. Модель визначає групу об’єктів, значення їх атрибутів і залежності між ними.

Головним параметром сутності в системі є унікальний ідентифікатор. Він присвоюється кожному створеному об’єкту і допомагає розрізнити його в групі ідентичних об’єктів.

3.2.1 Модель Advertisement

Сутність «Оголошення» - ключова сутність системи. Користувач може створювати власні оголошення, а також знаходити оголошення, що його цікавлять.

Під час аналізу та проектування системи, було визначено, що сутність «Оголошення» повинна містити в собі наступні компоненти:

- Унікальний ідентифікатор.
- Інформацію про оголошення: дата створення, кількість переглядів тощо.
- Інформація про транспортний засіб.
- Інформація про користувача, який створив оголошення.

Для відображення сутності в програмному інтерфейсі було створено клас Advertisement, опис якого зображено на рисунку 3.1.

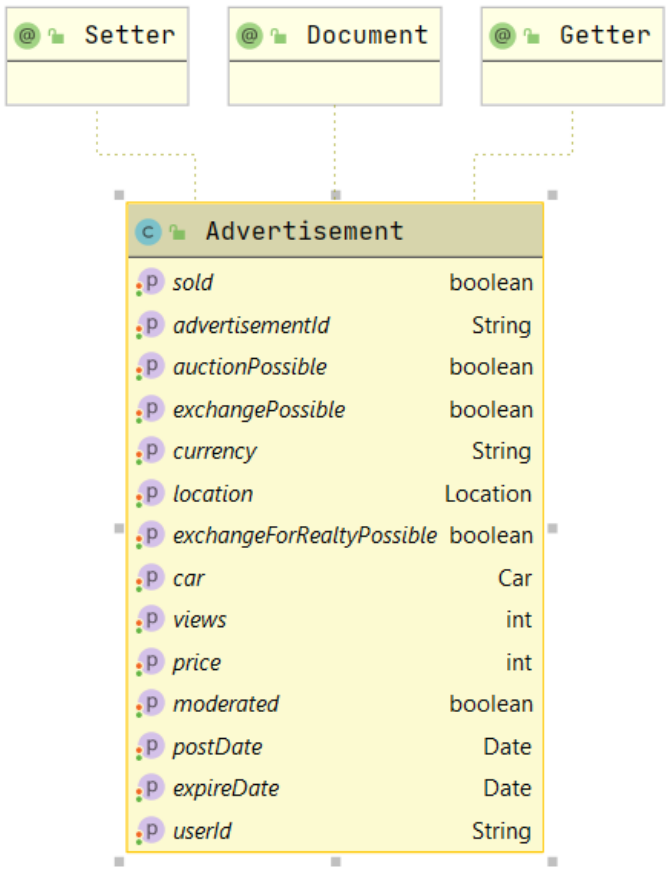


Рисунок 3.1. Клас Advertisement

Над описом класу позначені Java-анотації, що містять в собі метадані, необхідні компілятору для успішної компіляції і правильної роботи додатку:

Setter – анотація, що належить до бібліотеки Lombok. Під час компіляції програмного коду створює set-метод (сеттер) для кожного поля класу. Дозволяє уникнути написання зайвого шаблонного коду.

Document – анотація використовується фреймворком Spring Data для розпізнавання класу, як сутності MongoDB. Тобто, вона вказує на те, яку структуру має мати документ в колекції.

Getter – аннотація, ідентична Setter, про те вона вказує на необхідність створення get-метода (геттера) для кожного поля.

Клас має такі поля:

- Sold – приймає булеве значення, якщо true – оголошення знайшло покупця для автомобіля і він був проданий, інакше – оголошення ще відкрите.

- AdvertisementId – унікальний ідентифікатор оголошення, генерується MongoDB при створення документа в колекції.
- AuctionPossible – булеве поле, що вказує на можливість торгу під час покупки.
- ExchangePossible – булеве поле, яке описує можливість обміну транспортного засобу, що вказаний в оголошенні на інший.
- Currency – валюта, в якій вказана ціна на автомобіль. Може набувати лише трьох значень: українська гривня, долар США, євро.
- Location – місце розташування транспортного засобу. Посилання на іншу сутність.
- ExchangeForRealtyPossible – булеве поле, якщо true – можливий обмін на нерухомість, інакше – ні.
- Car – поле, що містить дані про транспортний засіб. Посилання на іншу сутність.
- Views – цілочисельний лічильник переглядів оголошення.
- Price – ціна транспортного засобу.
- Moderated – булеве поле, якщо true – оголошення вже пройшло модерацію і може бути розміщене в системі, інакше – оголошення ще на розгляді операторів.
- PostDate – дата та час розміщення оголошення в часовому поясі користувача прикладного програмного інтерфейсу.
- ExpireDate -дата, до якої оголошення буде розміщено в системі. Завжди становить рівно півроку від дати розміщення.
- UserId – унікальний ідентифікатор користувача системи, котрий створив та розмістив оголошення в системі.

3.2.2 Модель Car

Сутність «Автомобіль» представляє собою транспортний засіб, який виставили на продаж. «Автомобіль» залежить від сутності «Оголошення» за принципом «один до одного», тобто один конкретний автомобіль може бути розміщений лише в одному оголошенні, а одне оголошення може описувати лише один автомобіль.

Для опису сутності в програмному коді було створено клас Car, описаний на рисунку 3.2.

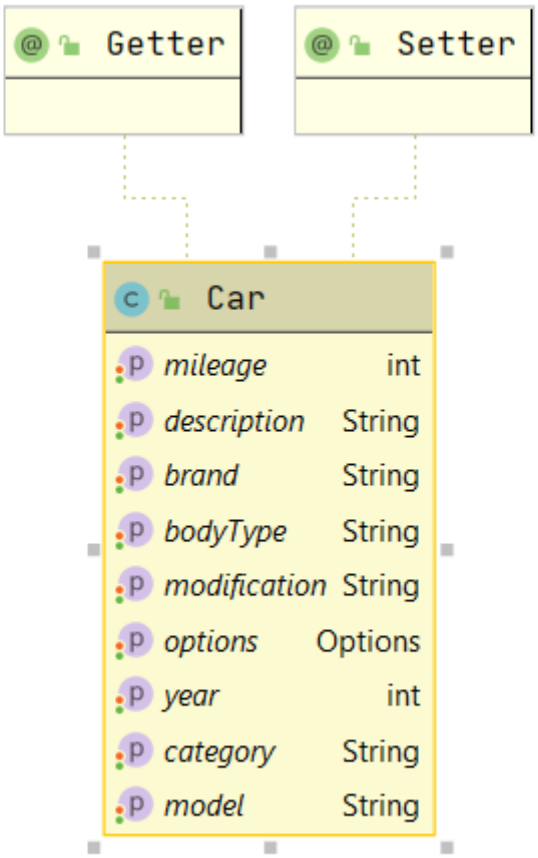


Рисунок 3.2. Клас Car

Модель «автомобіль» описана в класі такими полями:

- Mileage – цілочисельне значення, описує пробіг автомобіля в кілометрах.
- Description – опис транспортного засобу від власника.

- Brand – назва виробника автомобіля.
- BodyType - тип кузова автомобіля: седан, хетчбек, універсал тощо.
- Modification – модифікація автомобіля.
- Options – поле описує додаткові опції автомобіля. Посилання на іншу сутність.
- Year – рік випуску автомобіля.
- Category – категорія транспортного засобу: легковий, вантажний, комерційний тощо.
- Model – конкретна модель виробника автомобіля.

Сутність «автомобіль» не має унікального ідентифікатора, оскільки автомобіль певного виробника та марки не може бути унікальним.

3.2.3 Модель Location

Сутність «місцеположення» репрезентує локацію транспортного засобу вказаного в оголошенні. Сутність «оголошення» має посилання на сутність «місцеположення». «Місцеположення» було винесено в окрему сутність задля збереження абстрактного підходу до компонентів системи.

В прикладному програмному інтерфейсі «місцеположення» описано завдяки класу Location, що зображений на рисунку 3.3.

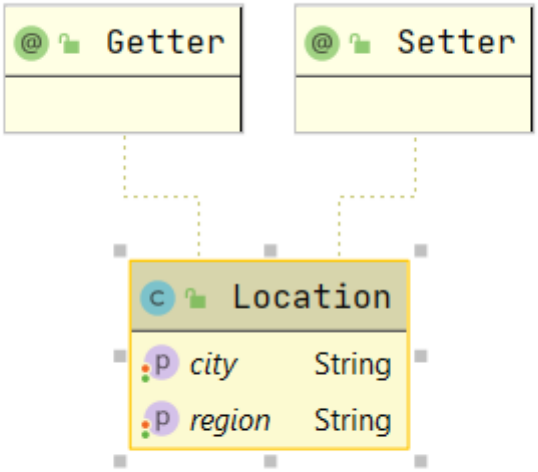


Рисунок 3.3. Клас Location

Клас Location містить в собі лише два поля:

- City – назва населеного пункту України: обласний або районний центр.
- Region – назва області України, до якої відноситься населений пункт, що вказаний в параметрі city.

3.2.4 Модель Options

Сутність «опції» відображає другорядні властивості автомобіля, які не є важливими для створення оголошення, проте може більше детально описати транспортний засіб потенційним покупцям. Наприклад, наявність кондиціонеру, клімат-контролю, типу коробки перемикання передач.

Завдяки вказаним параметрам сутності «опції», користувачі зможуть більш детально виконувати пошук транспортних засобів, зменшуючи діапазон пошуку.

Для відображення «опції» в системі, що розробляється в рамках даної бакалаврської роботи, було створено клас Options, опис якого відображений на рисунку 3.4.

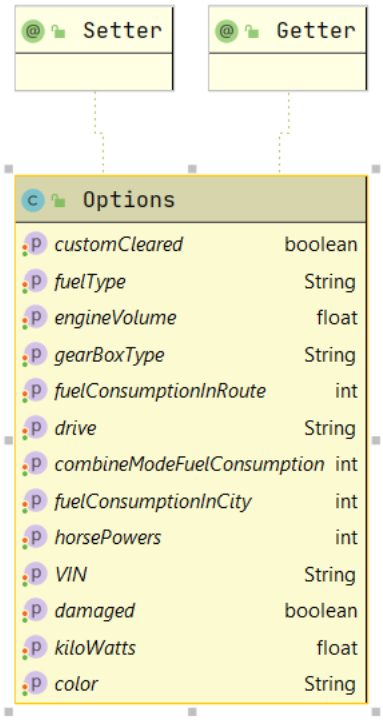


Рисунок 3.4. Клас Options

Під час проектування прикладного програмного інтерфейсу було вирішено надати класу Options такі поля:

- CustomCleared – булеве поле, якщо true – автомобіль розмитнений, інакше – на іноземній реєстрації.
- FuelType – тип палива, який споживає автомобіль. Має обмежений перелік значень: бензин, дизель, електрика, пропан-бутан, метан, газ\бензин.
- EngineVolume – робочий об’єм двигуна внутрішнього згорання.
- GearBoxType – тип коробки перемикання передач. Може бути або механічна, або автоматична.
- FuelConsumtionInRoute – цілочисельне значення, що відображає об’єм пального в літрах, витраченого в русі поза населеним пунктом за 100 кілометрів шляху.
- Drive – тип приводу трансмісії транспортного засобу. Має три значення: передній, задній, повний.
- CombineModeFuelConsumption – цілочисельне значення, що відображає об’єм пального в літрах, що було витрачене на рух в комбінованому режимі за 100 кілометрів шляху.
- FuelConsumptionInCity – цілочисельне значення, що відображає об’єм пального в літрах, що було витрачене, рухаючись в межах населеного пункту за 100 кілометрів шляху.
- HorsePowers – потужність двигуна в кінських силах.
- VIN – (англ. Vehicle identification number) – унікальний серійний номер, що використовується в автомобільній промисловості для ідентифікації механічного транспортного засобу.
- Damaged – булеве поле, якщо true – автомобіль був учасником дорожньо-транспортної пригоди.
- KiloWatts - потужність двигуна в кіловатах.
- Color – назва кольору кузова транспортного засобу.

3.3 Загальна модель оголошення

Після створення загальної моделі оголошення та її репрезентації в програмному коді, було створено колекцію advertisements в базі даних MongoDB.

Завдяки можливостям фреймворку Spring Data та документо-орієнтованому підходу СУБД MongoDB немає потреби створювати колекції самостійно. Spring Data надсилатиме в колекцію MongoDB дані в такому вигляді, як вони описані в класі Advertisement, завдяки чому всі документи матимуть однакову структуру. Створення колекції також виконує фреймворк Spring Data, достатньо лише вказати потрібну назву колекції і вона буде створена при першій спробі збереження об'єкту в базу даних.

Загальна модель сутності «оголошення» та з залежними сутностями описана на рисунку 3.5. Створеної моделі більш ніж достатньо для реалізації базового функціоналу прикладного програмного інтерфейсу.

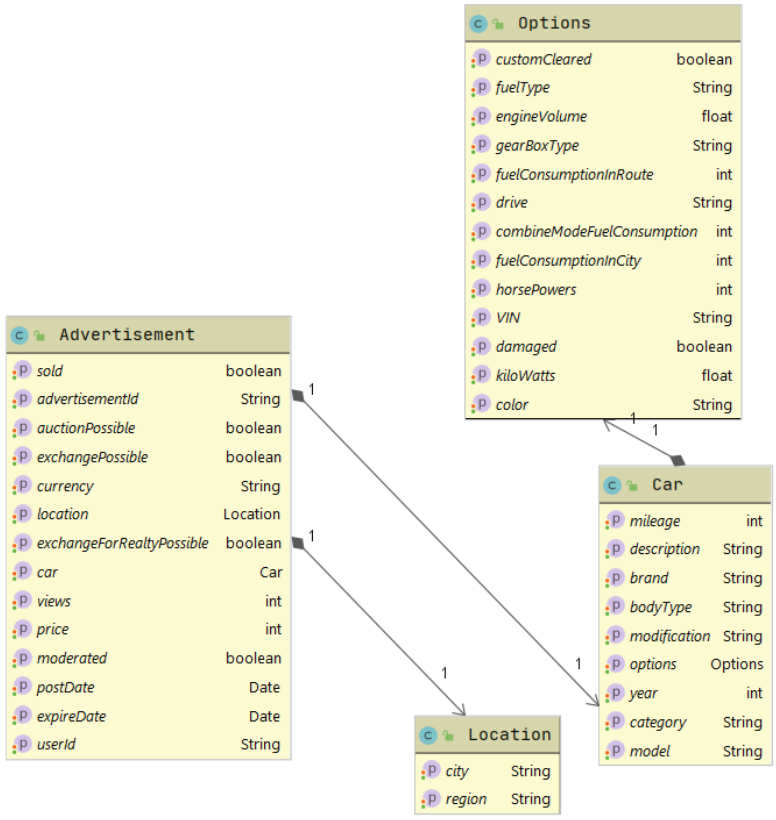


Рисунок 3.5. Загальна модель оголошення

3.4 Функціонал прикладного програмного інтерфейсу

Основні операції прикладного програмного інтерфейсу базуються на функціях керування даними: створення, оновлення, видалення та вибір даних з джерела інформації. В API для веб-застосунків ці операції реалізовуються з використанням HTTP-протоколу та його стандартних методів. Використання методів HTTP дозволяє клієнту звертатись до прикладного програмного інтерфейсу за однією і тією ж URL-адресою, проте виконувати різні операції.

HTTP-протокол виконує роль мосту між клієнтом та сервером, передаючи дані користувача та інформацію про те, яку дію слід виконати серверу. Після отримання запиту і його успішної обробки, сервер виконує операцію з базою даних, використовуючи функцію керування даними. В залежності від того чи успішно була виконана операція чи ні, клієнт отримує чітко визначену відповідь від сервера, яка дає йому можливість визначити статус обробки запиту сервером.

В прикладному програмному інтерфейсі, що створюється в рамках даного дипломного проекту використовується HTTP протокол версії 1.1. Структура програмного продукту представляє собою трирівневу архітектуру: об’єкт доступу до даних; сервіс, що реалізовує бізнес-логіку API; контролер – точка прийому даних від користувача.

3.5 Об’єкт доступу до даних оголошень

Для реалізації CRUD-операції був створений DAO-об’єкт, що надає абстрактний інтерфейс для доступу до даних. В реалізації власного DAO використовується фреймворк Spring Data, що надає високорівневий рівень доступу до бази даних, зокрема для MongoDB. Реалізація власного DAO-об’єкта надає користувачеві доступ до бази даних, що використовується прикладним програмним інтерфейсом без розкриття деталей самої бази даних.

Для роботи з оголошеннями було створено інтерфейс AdvertisementDao, що зобов’язує розробника реалізувати бізнес-логіку операцій над оголошеннями і надає можливість заміни однієї реалізації DAO на іншу,

оскільки в проекті використано механізм впровадження залежностей.

Прикладний програмний інтерфейс для системи продажу автомобілів надає власну стандартну реалізацію для виконання CRUD-операцій над оголошеннями. Для цього було створено клас під назвою AdvertisementDaoImpl, що реалізовує інтерфейс AdvertisementDao. Зміст класу зображено на рисунку 3.6.

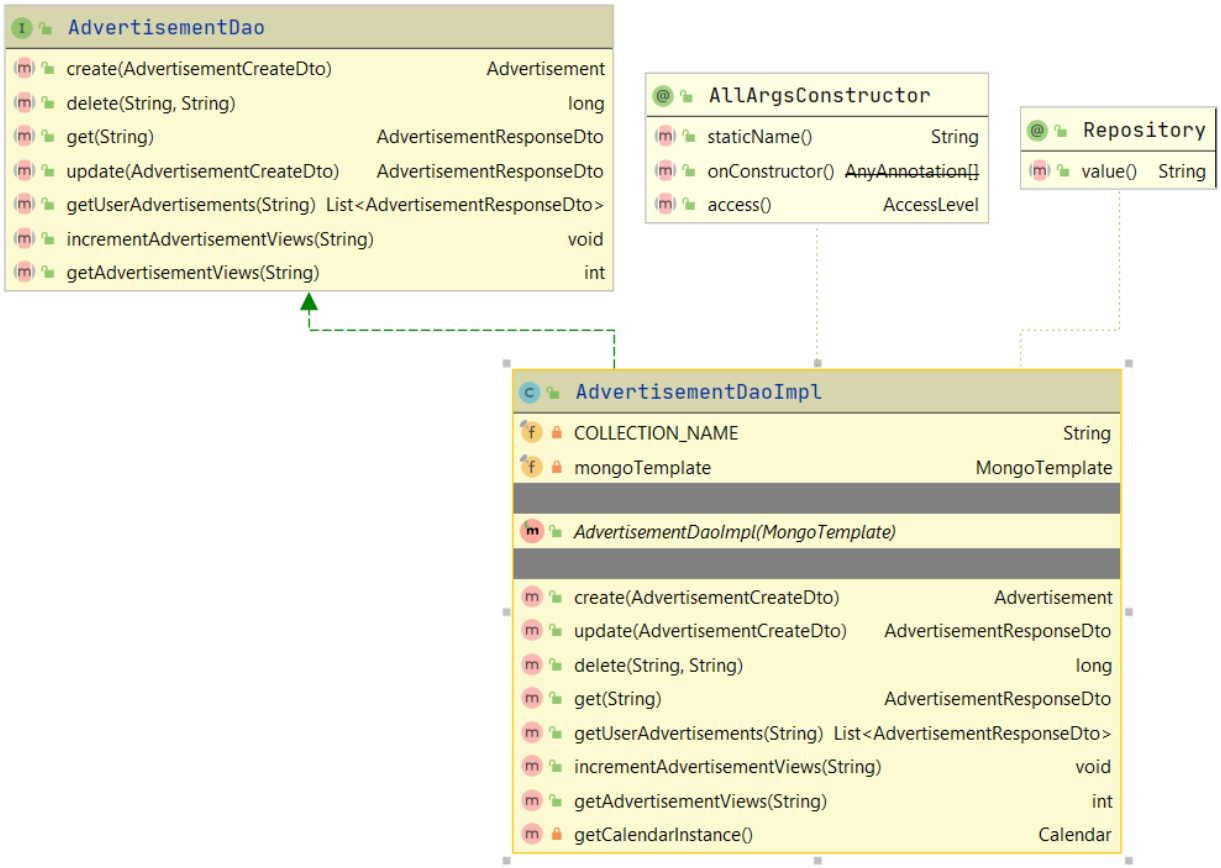


Рисунок 3.6. UML-діаграма для AdvertisementDao та його реалізації

CRUD-операції для оголошень в AdvertisementDao описані такими методами: create, update, delete, get.

В класі AdvertisementDaoImpl ці методи реалізовані таким чином:

- create – приймає як аргумент об’єкт типу AdvertisementCreateDto, який представляє дані, що надіслав клієнт. Дані з об’єкту типу AdvertisementCreateDto копіюються в об’єкт типу Advertisement. Використовуючи метод insert, що представлений класом

MongoTemplate фреймворку Spring Data, оголошення зберігається в колекцію MongoDB. Метод повертає опис створеного оголошення.

- get – описує операцію отримання даних про оголошення з колекції MongoDB, використовуючи унікальний ідентифікатор створеного в базі даних оголошення.
- delete – метод реалізовує операцію видалення оголошень певного користувача з бази даних. Для цього спочатку виконується пошук записів в базі даних оголошень за унікальними ідентифікаторами користувача і оголошення. Якщо такі записи знайдені, вони видаляються.
- update – метод отримує запит від клієнта, що ідентичний запиту на створення оголошення, проте містить в собі нові значення певних параметрів. Якщо за ідентифікатором оголошення було знайдено запис в базі даних, то значення параметрів змінюються на нові та зберігаються. У відповідь користувач отримує сутність оголошення з новими значеннями.

Даний DAO-об’єкт використовує впровадження залежностей через конструктор. Під час компіляції програмного коду IoC-контейнер фреймворку Spring додає залежність MongoTempate.

Java-анотація AllArgsConstructor з бібліотеки Lombok створює конструктор класу не етапі компіляції, що дозволяє не писати зайвий шаблонний код і робить структуру класу більш зручною для читання розробником. Анотація Repository використовується фреймворком Spring, для того, щоб визначити анотований клас як репозиторій. Цей підхід дозволяє отримати прозору трансляцію виключень, що виникають під час помилок, які пов’язані з роботою з базою даних. Завдяки цьому, сервіс, що буде використовувати даний об’єкт DAO та обробляти виключні ситуації.

Об’єкт DAO є найнижчим рівнем трирівневої структури прикладного програмного інтерфейсу і відповідає за безпосереднє виконання функцій керування даними.

У випадку необхідності створення іншого DAO-об’єкта розробнику достатньо створити нову реалізацію інтерфейсу AdvertisementDao, визначити в методах власну логіку та позначити об’єкт, щоб механізм впровадження залежностей міг використати нову реалізацію.

3.6 Сервіс для роботи з оголошеннями

На другому рівні архітектури прикладного програмного інтерфейсу для системи продажу автомобілів реалізований сервіс для роботи з оголошеннями. Сервіс – це компонента програмного інтерфейсу, що реалізує бізнес-логіку додатку. Сервісний рівень допомагає відокремити бізнес-логіку додатку та логіку роботи з даними. В такому випадку, зміна бізнес-логіки не буде впливати на механізм роботи рівня доступу до даних, і навпаки, зміна структури даних або способу їх збереження не впливатиме на бізнес-логіку, за умови правильної реалізації інтерфейсу.

Для реалізації бізнес-логіки роботи з оголошеннями було створено інтерфейс AdvertisementService та його реалізацію AdvertisementServiceImpl. Структура сервісу описана на рисунку 3.7.

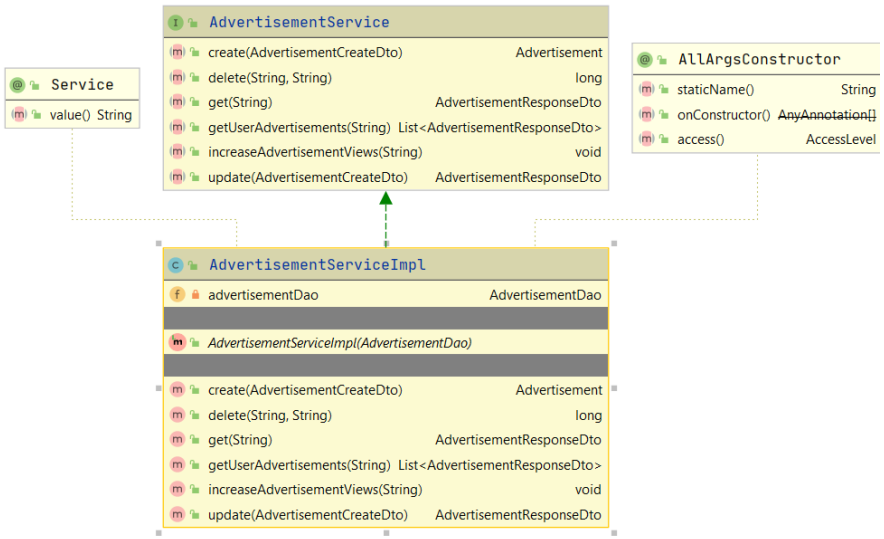


Рисунок 3.7. Сервіс AdvertisementService та його реалізація

Використовуючи механізм впровадження залежностей, сервіс для роботи з оголошеннями отримує поточну реалізацію об’єкту для роботи з даними та використовує методи DAO у своїй реалізації.

Наприклад:

- Метод create сервісу AdvertisementServiceImpl виконує перевірку на те, чи прийшли дані від користувача. Якщо жодних даних не було отримано, то метод create DAO-об’єкта не викликається і повертається значення null.
- Метод delete сервісу виконує операцію видалення лише в тому випадку, якщо користувач надав всі параметри: унікальний ідентифікатор оголошення і користувача. В такому випадку буде викликаний метод delete DAO-об’єкта, який поверне кількість видалених оголошень з бази даних MongoDB. Якщо операція видалення не була виконана, метод сервісу повертає значення -1, що сигналізує про невдалу спробу видалення даних.

Анотація Service використовується для того, щоб фреймворк Spring міг розпізнати клас як компоненту сервісного рівня додатку.

3.7 Точка прийому даних від клієнта

Точка прийому даних (англ. endpoint) – компонент прикладного програмного інтерфейсу, що знаходиться на третьому, останньому рівні архітектури додатку та відповідає за отримання даних від користувача та відправку відповіді в залежності від результату виконаної операції. В рамках розробки API точки прийому також називають контролерами.

Клас, що виконує функції контролера має бути позначеним анотацією Controller або RestController, що дозволяє фреймворку Spring ідентифікувати компоненту як точку прийому даних.

Для того, щоб контролер міг приймати дані, йому необхідно визначити URL-адресу, яка передається як параметр анотації RequestMapping.

Структура класу AdvertisementController, що виконує CRUD-операції для оголошень, зображена на рисунку 3.8.

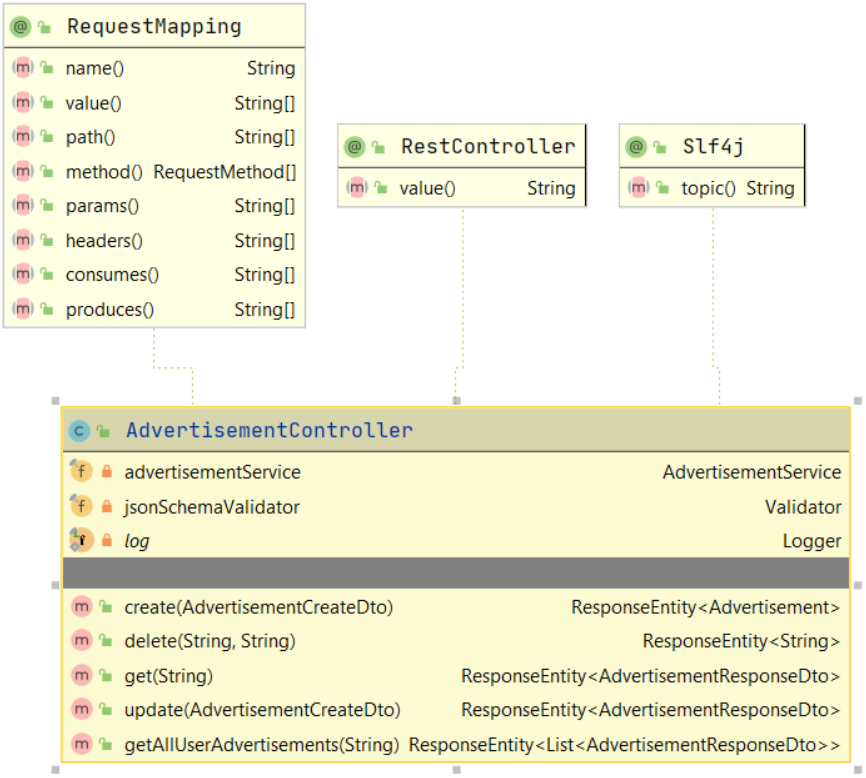


Рисунок 3.8. Контролер для роботи з оголошеннями

3.7.1 Операція створення оголошення

Для того, щоб створити нове оголошення в системі, користувач має відправити POST-запит за URI {api-root}/advertisement/create, де api-root – це адреса сервера, на якому розгорнутий прикладний програмний інтерфейс. HTTP-метод POST використовується для передачі даних користувача заданому ресурсу. Дані, які користувач бажає відправити, повинні бути вказані в тілі запиту. Метод POST не вважається ідемпотентним, тобто повторення одних і тих самих запитів може повертати різні результати. Наприклад, після кожної спроби створити оголошення, буде створюватись копія, оскільки оголошення відрізняються один від одного завдяки унікальному ідентифікатору.

Також необхідно вказати в запиті заголовок Content-Type і присвоїти йому значення application/json. Цей заголовок дозволяє серверу зрозуміти, що користувач очікує отримати відповідь на запит в форматі JSON.

Для відправки запиту використовується додаток Postman, він має широкий функціонал для тестування прикладних програмних інтерфейсів та взаємодії з

ними. Тестовий запит в графічному інтерфейсі Postman зображений на рисунку 3.9.

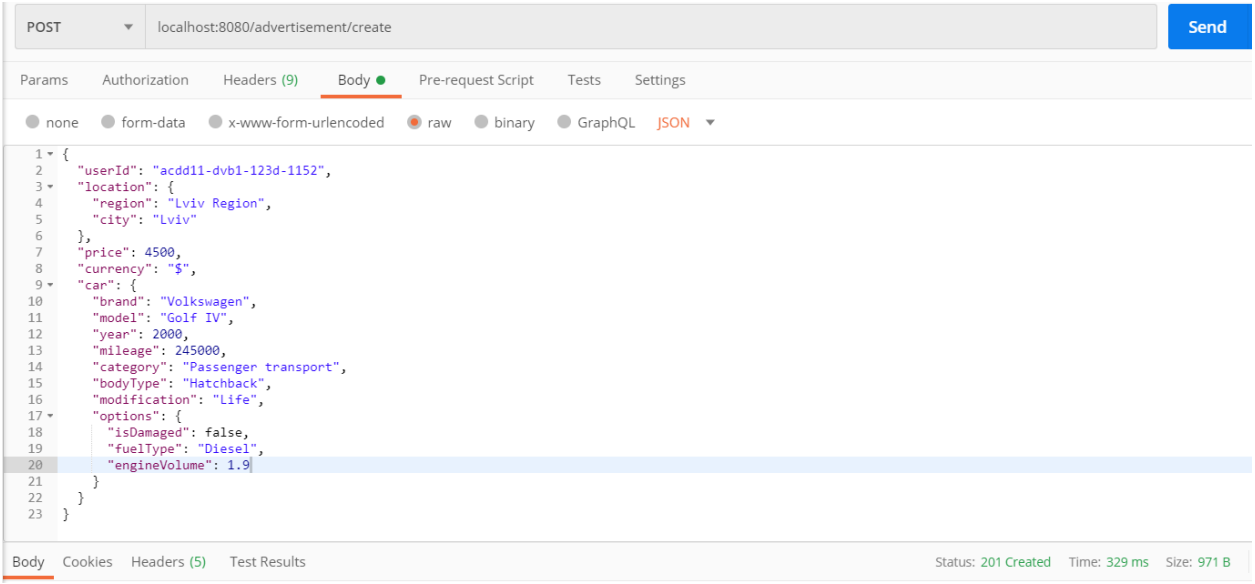


Рисунок 3.9. Приклад запиту для створення оголошення.

Оскільки, запит був успішно виконаний, то користувач отримав у відповідь опис створеного оголошення, а також HTTP-статус 201-Created.

Статус 201 означає успішне виконання запиту і повідомляє про те, що сутність була успішно створена в базі даних. Вигляд відповіді на запит зображений на рисунку 3.10. Дані відповіді було перенесено в текстовий редактор Notepad++ для кращого відображення.

```
{
    "advertisementId": "5ed7db92071a0c102e3eedf0",
    "userId": "acdd11-dvb1-123d-1152",
    "postDate": "2020-06-03T17:19:14.141+0000",
    "expireDate": "2020-12-03T18:19:14.141+0000",
    "views": 0,
    "location": {
        "region": "Lviv Region",
        "city": "Lviv"
    },
    "price": 4500,
    "currency": null,
    "car": {
        "brand": "Volkswagen",
        "model": "Golf IV",
        "year": 2000,
        "mileage": 245000,
        "category": "Passenger transport",
        "bodyType": "Hatchback",
        "modification": "Life",
        "options": {
            "gearBoxType": null,
            "drive": null,
            "fuelType": "Diesel",
            "fuelConsumptionInCity": 0,
            "fuelConsumptionInRoute": 0,
            "combineModeFuelConsumption": 0,
            "engineVolume": 1.9,
            "horsePowers": 0,
            "kiloWatts": 0.0,
            "color": null,
            "damaged": false,
            "customCleared": false,
            "vin": null
        },
        "description": null
    },
    "sold": false,
    "exchangePossible": false,
    "moderated": false,
    "auctionPossible": false,
    "exchangeForRealtyPossible": false
}
```


Рисунок 3.10. Відповідь на POST-запит

Всі необов’язкові параметри, що не були вказані в запиті за замовчуванням набувають значень 0, якщо це числові параметри та false – якщо булеве.

3.7.2 Операція вибору оголошення

Для того, щоб отримати оголошення за його унікальним ідентифікатором, необхідно виконати GET-запит за URI-ідентифікатором:

{api-root}/advertisement/get/advertisementId/{advertisementId}, де api-root – це адреса сервера, а advertisementId – значення унікального ідентифікатора оголошення.

GET-метод використовується для того, щоб отримати зміст вказаного ресурсу або розпочати певний процес на сервері. GET-запити вважаються ідемпотентними, тобто повторення одного і того ж запиту завжди матиме однаковий результат. Приклад GET-запиту для оголошення, що було створено

в розділі 3.7.1 зображено на рисунку 3.11.

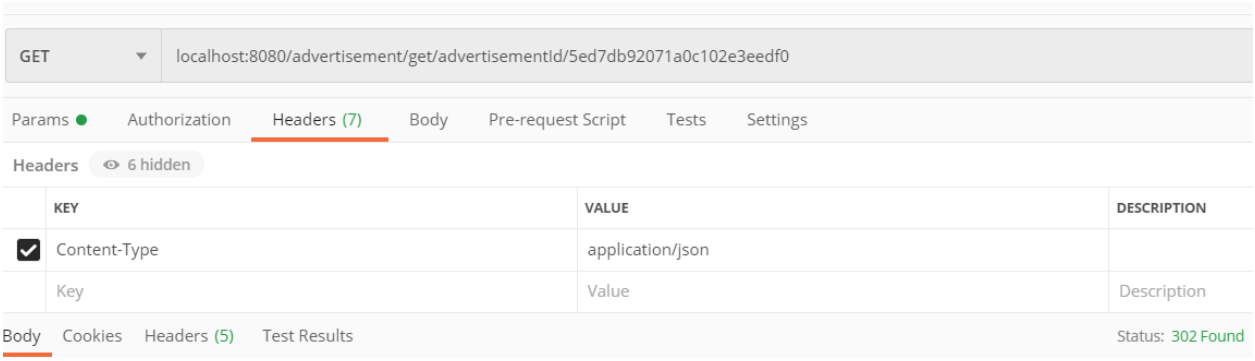


Рисунок 3.11. GET-запит в Postman

При успішному виконанні запиту, сервер повертає опис знайденого оголошення, а також HTTP-статус 302 - Found. Статус 302 повідомляє користувача, що вказаний ресурс було знайдено. Результат GET- запиту ідентичний, тому, що зображений на рисунку 3.10.

3.7.3 Операція видалення оголошення

Для того, щоб видалити оголошення з системи, необхідно відправити HTTP-запит з методом DELETE за URI-ідентифікатором:

{apiroot}/advertisement/delete/userId/{userId}/advertisementId/{advertisementId}

де userId – цей унікальний ідентифікатор користувача, а advertisementId – ідентифікатор оголошення.

Операція видалення буде виконана з використанням Postman (рисунок 3.12).

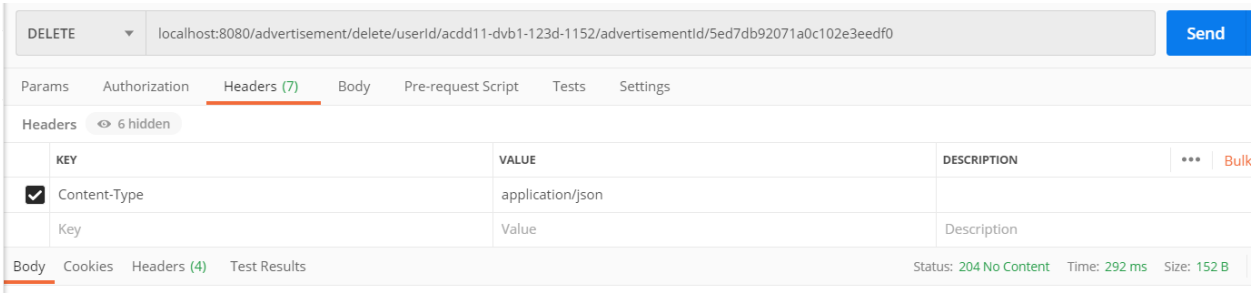


Рисунок 3.12 Приклад видалення оголошення

Операція видалення повертає лише HTTP-статус 204 – No Content, якщо запит був успішно виконаний. Статус 204 інформує користувача прокладного програмного інтерфейсу про те, що вказаний ресурс був видалений і більше не існує в системі.

3.7.4 Операція оновлення оголошення

Щоб виконати операцію оновлення певних параметрів оголошення, необхідно виконати PATCH-запит, вказавши URI:

{api-root}/advertisement/update, де api-root – це адреса сервера.

PATCH-метод завантажує певну частину ресурсу на сервер. Для успішного виконання операції оновлення оголошення, необхідно вказати дані в тілі запиту, аналогічно POST-запиту, який створює оголошення, проте містить нові значення параметрів. На рисунку 3.13 зображена конфігурація запиту в Postman.

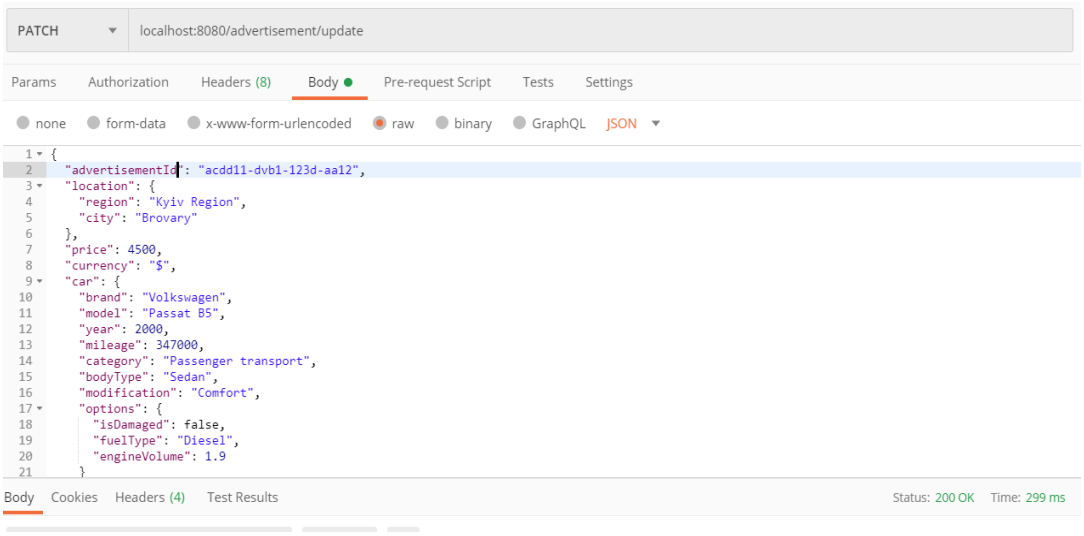


Рисунок 3.13. Операція оновлення оголошення

При успішному виконанні запиту, сервер повертає HTTP-статус 200 – OK. Це стандартний код для методів HTTP-протоколу, що інформує про успішне виконання запиту. Нові значення параметрів були збережені в документі MongoDB сутності «оголошення».

Варто зазначити, що унікальний ідентифікатор оголошення обов'язково повинен бути вказаний в тілі запиту, інакше виникне помилка, пов'язана з тим, що DAO-об'єкт не розпізнає який саме документ в базі даних йому потрібно оновити.

3.8 Операція пошуку оголошень за вказаними параметрами

Операція пошуку переліку оголошень не відноситься до набору CRUD-операцій, проте є дуже важливою функцією прикладного програмного інтерфейсу. Пошук виконується при надсиланні GET-запиту за

ідентифікатором URI {api-root} /search/advertisements?param1&…&paramN, де api-root – адреса сервера, param1&…&paramN – параметри GET- запиту.

Кожен параметр запиту відповідає параметру пошуку оголошення: ціна, марка авто, пробіг тощо. Параметри розділяються між собою амперсандом.

Приклад успішної операції пошуку зображено на рисунку 3.14.

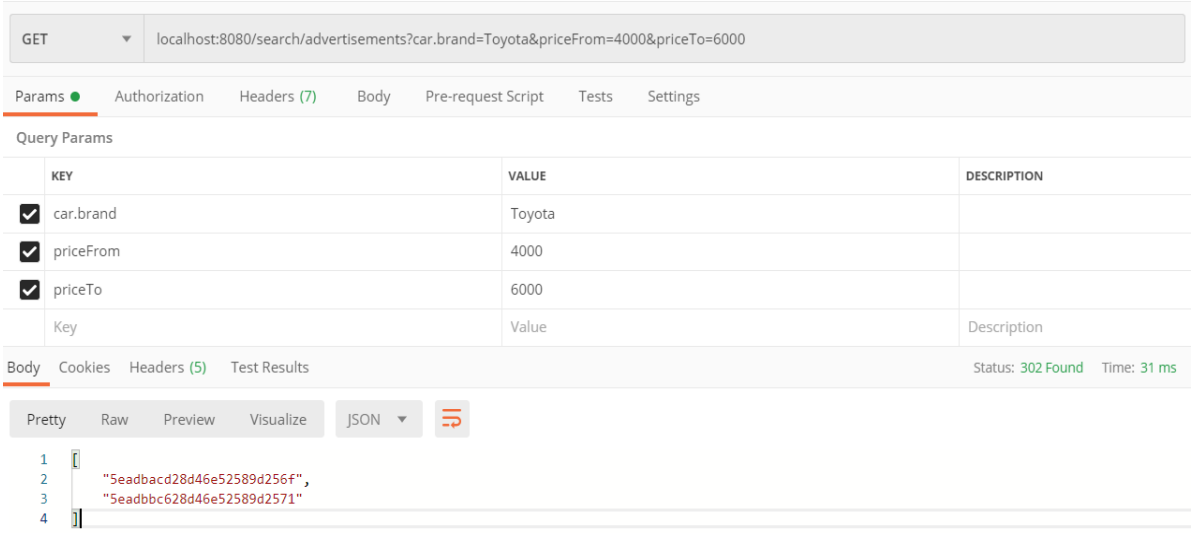


Рисунок 3.14. Приклад операції пошуку оголошень

Якщо запит був виконаний успішно, сервер поверне JSON-об’єкт, який вміщує масив унікальних ідентифікаторів оголошень, що задовольняють вказані в запиті критерії. В даному прикладі був виконаний пошук оголошень про продаж автомобілів бренду Toyota в ціновій категорії від 4 до 6 тисяч доларів США. В локальній базі даних було знайдено два таких документи. Також сервер повертає HTTP-статус 302 – Found. Код 302 інформує про те, що шукані оголошення були знайдені.

ВИСНОВОК ДО РОЗДІЛУ 3

В третьому розділі даної бакалаврської роботи було описано програмну архітектуру прикладного програмного інтерфейсу для системи продажу автомобілів. Програмний інтерфейс побудований за принципом трирівневої архітектури, в якій виділяються такі рівні: рівень роботи з даними, рівень бізнес-логіки, рівень контролерів. Кожен з цих рівнів детально описаний в розділі.

Також було показано модель сутностей прикладного програмного інтерфейсу, які на рівні абстракцій відображають об’єкти з реального світу в програмному коді.

В розділі описана реалізація основних функцій управління даними або CRUD-операцій, кожна операція доступна користувачеві через окрему URL-адресу. Для передачі даних від користувача до сервера використовується HTTP-протоколі, а зокрема методи: GET, POST, DELETE та PATCH. Для інформування користувача про успішно виконаний або провалений запит використовуються HTTP-статуси.

ВИСНОВКИ

В рамках виконання даної бакалаврської дипломної роботи було розглянуто принципи побудови сучасних прикладних програмних інтерфейсів для веб-застосунків, зокрема архітектурному підходу REST, який дозволяє проектувати та створювати прикладні програмні інтерфейси, які з часом просто підтримувати та масштабувати.

Загальноприйнятим є використання в REST API протоколу HTTP, як головного протоколу передачі даних між клієнтом та сервером. Дані передаються у форматі JSON, що дозволяє структуровано і лаконічно описати запит до сервера і відповідь сервера клієнту.

Оскільки, метою даного дипломного проекту було створення спеціалізованого прикладного програмного інтерфейсу для використання його в системі продажу автомобілів, було зібрано і проаналізовано значну кількість інформації щодо тенденцій, проблем бізнесу, певних підходів, що притаманні сфері продажу автомобілів та супутніх товарів. Значну увагу було приділено категорії малого та середнього бізнесу, який в майбутньому міг би користуватись розробленим прикладним програмним інтерфейсом та вирішувати з його допомогою проблеми ведення бізнесу.

Під час роботи було розроблено модель побудови програмного продукту, його структуру, взаємодії абстрактних сутностей, схему бази даних та її компонентів.

Зокрема, в першому розділі проекту було розглянуто аналоги систем для продажу автомобілів, порівняно можливості їх клієнтського графічного інтерфейсу та функціональні можливості.

Другий розділ описує технології, що були використані в процесі розробки програмного продуту.

Третій розділ описує основні аспекти прикладного програмного інтерфейсу для системи продажу автомобілів, який на даному етапі реалізовує функціонал для роботи з оголошеннями, а також модель сутностей системи.

Оскільки, сутність «оголошення» є фундаментальною в системі продажу, було створено прикладний програмний інтерфейс для взаємодії з цими сутностями. Користувач отримує реалізацію основних функцій управління даними, які відомі як CRUD-операції. Також користувач має можливість виконувати пошук оголошень в системі за заданими критеріями та отримувати у відповідь перелік унікальних ідентифікаторів шуканих оголошень.

Варто зазначити, що розроблений проєкт є перспективним, він може бути масштабований та реалізовувати новий функціонал для взаємодії з певними бізнес-сутностями, що дозволить споживачам програмного продукту отримувати максимальну користь та прибуток завдяки розробленому прикладному програмному інтерфейсу для систем продажу автомобілів.

Всі поставлені перед початком виконання роботи завдання та цілі були виконані. Мета дипломного проєкту досягнута.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Application Programming Interface [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Application_programming_interface
2. Мова програмування Java [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Java
3. Хорстманн К., «Java. Библиотека профессионала», «Діалектика» Київ 2020.
4. Java Virtual Machine [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Java_Virtual_Machine
5. Just-in-time compilation [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Java_Virtual_Machine
6. Ahead-of-time compilation [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Ahead-of-time_compilation
7. Фреймворк Spring [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Spring_Framework
8. Inversion of Control [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Inversion_of_control
9. СУБД MongoDB [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/MongoDB
10. Angular web-framework [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Angular_(web_framework)

ДОДАТОК 1

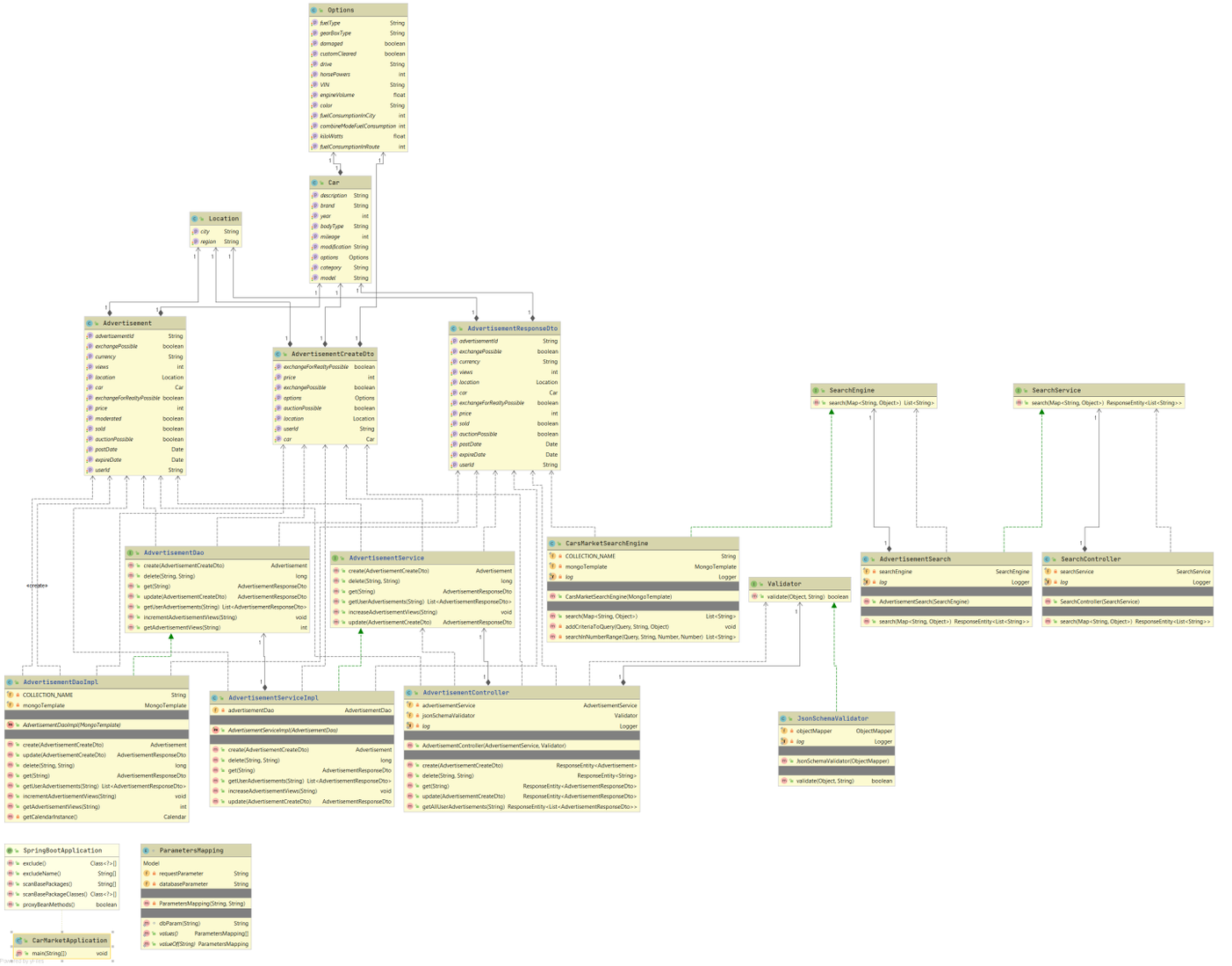
Прикладний програмний інтерфейс для системи продажу автомобілів

Схема структурна - діаграма класів

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Пилип'юк Д.О.			Схема структурна Діаграма класів	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				
						Арк. / Аркушів
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						

ДОДАТОК 2

Прикладний програмний інтерфейс для системи продажу автомобілів

Схема функціональна - блок-схема алгоритму

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020 р.

ІАЛЦ.467100.005 Д2

					ІАЛЦ.467100.005 Д2	
						Літ. / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата	Схема функціональна Блок-схема алгоритму	
Розроб.		Пилип'юк Д.О.				Арк. / Аркушів
Перевір.		Алещенко О.В.				
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						

ДОДАТОК З

Прикладний програмний інтерфейс для системи продажу автомобілів

Схема взаємодії

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ 2020 р.

					ІАЛЦ.467100.006 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Пилип'юк Д.О.			Схема взаємодії	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						